Title
Flexible Cross-Subspace Alignment Codes for Variable Coded Distributed Batch Matrix Multiplication

Author
Tauz, Lev

Publication Date
2020

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
Los Angeles

Flexible Cross-Subspace Alignment Codes for Variable Coded Distributed Batch Matrix Multiplication

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Science in Electrical and Computer Engineering

by

Lev Tauz

2020

ABSTRACT OF THE THESIS

Flexible Cross-Subspace Alignment Codes for
Variable Coded Distributed Batch Matrix Multiplication

by

Lev Tauz
Master of Science in Electrical and Computer Engineering
University of California, Los Angeles, 2020
Professor Lara Dolecek, Chair

Modern distributed systems suffer from a phenomenon known as stragglers where computation nodes either break-down or are sufficiently slow, resulting in a large tail latency. Inspired by error correcting codes, researchers within the field of coded computation combat stragglers by cleverly encoding the data within the computations. One major endeavor is in the study of coded matrix-matrix multiplication where the task is to multiply two large matrices in a distributed manner. Most coded matrix computation work focuses on highly structured tasks which allows for easier code construction and analysis but limits the applicability for more general problems. For the first time, we consider the novel problem of multiplying many different matrices whose products may share matrices with no guaranteed regularity. Specifically, we consider the Variable Coded Distributed Batch Matrix Multiplication (VCDBMM) problem where the system is given two sets of matrices $\mathcal{A} = \{A_1, A_2, \ldots, A_{|\mathcal{A}|}\}$ and $\mathcal{B} = \{B_1, B_2, \ldots, B_{|\mathcal{B}|}\}$ and a set of computation goals $\mathcal{S} = \{(i_1, j_1), (i_2, j_2), \ldots (i_{|\mathcal{S}|}, j_{|\mathcal{S}|})\}$ and the objective is to calculate the matrix multiplication $A_i B_j$ for every $(i, j) \in \mathcal{S}$ in the presence of stragglers. Therefore, a good coding solution minimizes the recovery threshold

(i.e., the number of workers that we need to wait for in order to compute the final output). Inspired by Cross-Subspace Alignment Codes, we construct Flexible Cross-Subspace Alignment Codes (FCSA) to solve the general VCDBMM problem. We provide two variants of FCSA codes that allow for a trade-off between the encoding/decoding complexity and the recovery threshold. We demonstrate that both variants are within a factor of two optimal under certain system constraints. We also generalize FCSA codes into Grouped FCSA codes where we group computations together to provide further flexibility between the computational complexity at the workers and the recovery threshold. We provide simulations on random instances of the VCDBMM problem and demonstrate the average improvement offered by our codes.

The thesis of Lev Tauz is approved.

Lin F. Yang

Achuta Kadambi

Lieven Vandenberghe

Lara Dolecek, Committee Chair

University of California, Los Angeles

2020

To my family -

who always hold me up when I need it most

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction and Motivation

Large scale distributed computation is a powerful modern tool that is used to tackle the exponential rise of big data. As the complexities of distributed systems grow, new bottlenecks and challenges arise. One such bottleneck is the presence of *stragglers* (i.e., workers that fail or are slow to respond) which significantly increases the tail latency of these distributed systems [4].

One natural method to alleviate stragglers is to add redundancy, such as replicating computations. Yet, simple replication incurs a prohibitively large resource overhead which necessitates the search for other strategies. In the field of distributed storage, it is well known that the technique of erasure coding provides robustness to failed nodes with minimal storage overhead [5]. In the same spirit, researchers apply erasure coding by encoding the data and treating stragglers as erasures. This endeavor has launched a whole new field of study termed "coded computation" [6]. The main idea behind coded computation is that workers perform computations on encoded inputs such that querying a sufficient subset of them allows for the recovery of the outputs from unreliable workers (i.e., stragglers). The minimum number of workers to query is known as the *recovery threshold*.

The field of coded computation was initially introduced for matrix-vector multiplication operations [6]. In the years after, a plethora of works developed various coded computation strategies applicable to linear matrix-vector computations [7–20] and bi-linear matrix-matrix computation [1–3, 21–28, 28–32]. While strategies for linear matrix-vector essentially apply classical error-correcting theory, coded computation on matrix-matrix computations requires

the development of brand new codes that take advantage of the unique and fundamental structure of the problem.

Codes for matrix-matrix computations are broadly separated into two categories: matrix partitioning for computing a single computation task [1, 21–23] and batch processing of multiple distinct computation tasks [2, 3, 28]. These codes provide resilience to stragglers while offering fast decoding techniques but ultimately rely on the rigid structure of the computation tasks they aim to compute. For example, Polynomial codes [1] are a matrix partitioning scheme that partition two matrices A and B into

$$A = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_n \end{bmatrix}, B = \begin{bmatrix} B_1 & B_2 & \cdots & B_m \end{bmatrix}$$

and aim to compute

$$AB = \begin{bmatrix} A_1B_1 & A_1B_2 & \cdots & A_1B_m \\ A_2B_1 & A_2B_2 & \cdots & A_2B_m \\ \vdots & \vdots & \vdots & \vdots \\ A_nB_1 & A_nB_2 & \cdots & A_nB_m \end{bmatrix}. \tag{1.1}$$

Polynomial codes achieve an optimal recovery threshold in computing all pair-wise computations of A_i and B_j needed to retrieve AB. Yet, what if the user did not want to know all pairwise computations but only a subset of them? While Polynomial codes could still compute the desired results, they require an unnecessarily large recovery threshold since not all output computations are necessary.

A similar issue arises for codes based on batch processing when the computation tasks share data [2,3]. These codes attempt to solve the problem of pairwise multiplying two sets of L matrices $\{A_1, A_2, \ldots, A_L\}$ and $\{B_1, B_2, \ldots, B_L\}$ into $\{A_1B_1, A_2B_2, \ldots, A_LB_L\}$. Similar to Polynomial codes, these sets of codes are known to be optimal for this specific computation

task. But what if the computation task required a product A_iB_j where $i \neq j$? There is a natural redundancy if, for example, a matrix A_i is required in multiple products. To our knowledge, all current batch-processing schemes do not utilize this redundancy in shared data.

In this work, we introduce the Variable Coded Distributed Batch Matrix Multiplication (VCDBMM) problem that generalizes the problem spaces of the previous two examples [1–3]. Assume a distributed system is provided with two sets of matrices $\mathcal{A} = \{A_1, A_2, \ldots, A_{|\mathcal{A}|}\}$ and $\mathcal{B} = \{B_1, B_2, \ldots, B_{|\mathcal{B}|}\}$ and a set of computation goals $\mathcal{S} = \{(i_1, j_1), (i_2, j_2), \ldots (i_{|\mathcal{S}|}, j_{|\mathcal{S}|})\}$ where the objective is to calculate the matrix multiplication A_iB_j for every $(i, j) \in \mathcal{S}$. Such a model can arise in many distributed computation tasks. For example, consider a system with many different users where each user has an associated A_i matrix of data. Each user wants to apply their data to a subset of $\mathcal{B}$ which may overlap with another user. As such, a good computation strategy should take full advantage of the natural redundancy offered in the task. Additionally, this strategy should also be resilient to stragglers by providing a small recovery threshold to better utilize the resources.

The major contribution of this work is the development of a new class of codes to efficiently solve the VCDBMM problem. We name these codes Flexible Cross-Subspace Alignment (FCSA) codes. The naming comes from how they are based on Cross-Subspace Alignment (CSA) codes [3]. We demonstrate two variants of FCSA codes that offer a trade-off between straggler resilience and encoding/decoding complexity. We show that both variants of FCSA codes are within a factor of two optimal under certain system constraints. In addition, we generalize the concept of FCSA codes to provide further flexibility between the computational complexity at the workers and the recovery threshold. We provide simulations on random instances of the VCDBMM problem to demonstrate the efficacy of using our codes instead of naively applying previous coding solutions.

The rest of this thesis is organized as follows. In Chapter 2, we formally introduce the VCDBMM model and provide relevant background on related code constructions. In

Chapter 3, we present the first variant of FCSA codes, which are named FCSA Type 1 (FCSA-T1) codes, that provide a factor of two optimal recovery threshold. In Chapter 4, we improve upon FCSA-T1 codes and develop FCSA Type 2 (FCSA-T2) codes that have a much smaller recovery threshold at the cost of higher encoding/decoding complexity. In Chapter 5, we improve the flexibility of FCSA codes by grouping computations tasks which provides flexibility in choosing the worker's computational complexity and the recovery threshold. Finally, the conclusion appears in Chapter 6.

Notation: Given a positive integer N, we define $[N]$ as the set $\{1, 2, 3, \ldots, N\}$. We define some common Big O notation to be used throughout this work. Let $f(x) = \mathcal{O}(g(x))$ indicate that there is a constant M and value x_0 such that $f(x) \leq Mg(x) \ \forall \ x \geq x_0$. Similarly, let $f(x) = \Omega(g(x))$ indicate that there is a constant k and value x_0 such that $kg(x) \leq f(x) \ \forall \ x \geq x_0$. The notation $\widetilde{\mathcal{O}}(a \log^2(b))$ suppresses polylog terms. It may be replaced with $\mathcal{O}(a \log^2 b)$ if the field supports the Fast Fourier Transform (FFT), and with $\mathcal{O}(a \log^2 b \log\log(b))$ if it does not.

CHAPTER 2

Preliminaries and Relevant Work

2.1 Variable Coded Distributed Batch Matrix Multiplication (VCDBMM)

We now properly define the VCDBMM problem. We follow similar notation as in [3]. As shown in Fig. 2.1, consider two source nodes each of which generates a set of matrices $\mathcal{A} = \{A_1, A_2, \ldots, A_{L_A}\}$ and $\mathcal{B} = \{B_1, B_2, \ldots, B_{L_B}\}$ such that for all $i \in [L_A]$ and $j \in [L_B]$ we have $A_i \in \mathbb{F}^{\alpha \times \beta}$ and $B_j \in \mathbb{F}^{\beta \times \gamma}$. We name these nodes as Source $\mathbb{A}$ and Source $\mathbb{B}$. Additionally, a set of computation goals is provided $\mathcal{S} = \{(i_1, j_1), (i_2, j_2), \ldots (i_{|\mathcal{S}|}, j_{|\mathcal{S}|})\}$ where the objective is for the fusion node to obtain the matrix multiplication $A_i B_j$ for every $(i, j) \in \mathcal{S}$. To accomplish this, the system has K worker nodes which perform the bulk of the computation. Each worker receives coded data from each source, performs some operation on them, and transmits the result to the fusion node. The fusion node then decodes the desired computations $\mathcal{S}$ from the responses of a subset of workers.

We now formalize the model in Fig. 2.1 using the following terminology. Let $\mathbf{f} = (f_1, f_2, \ldots, f_K)$ be the set of encoding functions for Source $\mathbb{A}$ and let $\mathbf{g} = (g_1, g_2, \ldots, g_K)$ be the set of encoding functions for Source $\mathbb{B}$. For the k^{th} worker, the source nodes transmit

$$\widetilde{A}_k = f_k(\mathcal{A}), \widetilde{B}_k = g_k(\mathcal{B}). \tag{2.1}$$

where $\widetilde{A}_k \in \mathbb{F}^{\widetilde{\alpha}_k \times \widetilde{\beta}_k}$ and $\widetilde{B}_k \in \mathbb{F}^{\widetilde{\beta}_k \times \widetilde{\gamma}_k}$.

We assume that workers are not very powerful and can only do simple tasks such as matrix-matrix multiplication. Therefore, the k^{th} worker can only multiply the two encoded

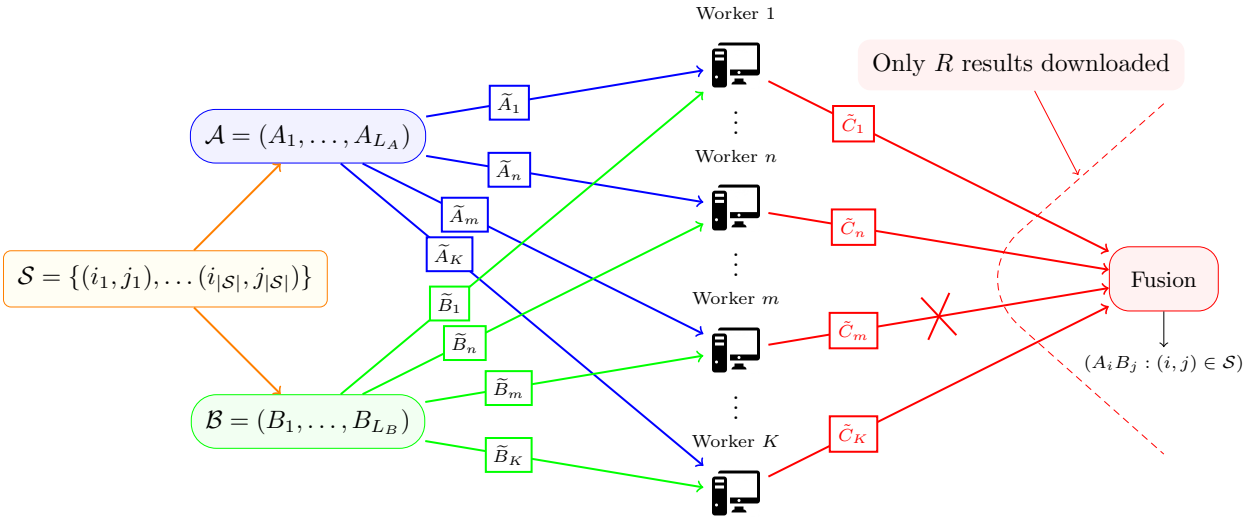


Figure 2.1: System Model for Variable Coded Distributed Batch Matrix Multiplication. There are two source nodes which produce two sets of matrices $\mathcal{A}$ and $\mathcal{B}$ and a set of computation goals $\mathcal{S}$. Each worker receives a coded matrix from each source and only computes the products of these matrices. Due to stragglers, only R outputs are received at the Fusion node where R is the recovery threshold.

matrices it receives and outputs

$$\widetilde{C}_k = \widetilde{A}_k \widetilde{B}_k \tag{2.2}$$

to the fusion node where $\widetilde{C}_k \in \mathbb{F}^{\widetilde{\alpha}_k \times \widetilde{\gamma}_k}$. This model assumes that some workers are stragglers and may fail to respond. The fusion node downloads the responses from the non-straggling workers and attempts to recover the desired products $\{A_i B_j : (i,j) \in \mathcal{S}\}$ using a class of decoding functions (denoted $\mathbf{d}$). We define

$$\mathbf{d} = \{d_{\mathcal{R}} : \mathcal{R} \subset [K]\}, \tag{2.3}$$

where $d_{\mathcal{R}}$ is the decoding function used when the set of non-straggling workers is $\mathcal{R} \subset [K]$. Therefore, we denote a VCDBMM code by the triple $(\mathbf{f}, \mathbf{g}, \mathbf{d})$.

A VCDBMM code is r-recoverable if the desired computations can be retrieved from the responses of at least r workers. Formally, a computation strategy $(\mathbf{f}, \mathbf{g}, \mathbf{d})$ is r-recoverable if

for any $\mathcal{R} \subset [K]$ such that $|\mathcal{R}| \geq r$ and for any realization of $\mathcal{A}$ and $\mathcal{B}$, then

$$d_{\mathcal{R}}(\{\widetilde{C}_i : i \in \mathcal{R}\}) = \{A_i B_j : (i,j) \in \mathcal{S}\}. \tag{2.4}$$

We define the recovery threshold R of a VCDBMM code as the minimum integer r such that the strategy is r-recoverable.

The communication cost that any VCDBMM code incurs is the upload cost of transmitting symbols from the sources to the workers and the download cost of transmitting symbols from the workers to the fusion node. Let U_A and U_B be the total number of symbols transmitted for the matrices $\widetilde{A}$ and $\widetilde{B}$ normalized by the number of workers, respectively, which we define as

$$U_A = \frac{1}{K} \sum_{i=1}^{K} \widetilde{\alpha}_k \widetilde{\beta}_k, \tag{2.5}$$

$$U_B = \frac{1}{K} \sum_{i=1}^{K} \widetilde{\beta}_k \widetilde{\gamma}_k. \tag{2.6}$$

We see that this is simply the average number of symbols sent to all workers. We treat U_A and U_B as the upload costs for this model.

Additionally, let D_C denote the total number of symbols transmitted into the fusion node before decoding, normalized by the number of messages. Therefore, we write D_C as

$$D_C = \max_{\mathcal{R}, \mathcal{R} \subset [K], |\mathcal{R}| = R} \frac{\sum_{k \in \mathcal{R}} \widetilde{\alpha}_k \widetilde{\gamma}_k}{R}, \tag{2.7}$$

where R is the recovery threshold. We treat D_C as the download cost for this model.

We note that the computational complexity at each worker is simply the number of arithmetic operations it takes to multiply two matrices. We assume that a simple matrix multiplication algorithm is used to multiply a $\widetilde{\alpha_k} \times \widetilde{\beta}_k$ matrix by a $\widetilde{\beta}_k \times \widetilde{\gamma}_k$ which has a complexity of $\mathcal{O}(\widetilde{\alpha_k} \widetilde{\beta}_k \widetilde{\gamma}_k)$. One may replace the simple algorithm for many fast matrix multiplication algorithms [33–36] without affecting the rest of the model. Since this can also be done for other coded computation schemes, applying these fast algorithms to our model

does not demonstrate any relative improvement. Thus, the simple algorithm is sufficient for our purposes.

We define a VCDBMM code as *shape-preserving* if encoding and decoding operations do not change the shape of the input matrices, i.e., $\widetilde{\alpha_k} = \alpha$, $\widetilde{\beta}_k = \beta$, and $\widetilde{\gamma}_k = \gamma$ for all $k \in [K]$. Implicitly, this guarantees that the computational complexity at each worker is $\mathcal{O}(\alpha\beta\gamma)$. Many codes fall under the shape-preserving constraint which we use as a common point to allow for a fair comparison between our VCDBMM codes and other relevant codes. Given the computation goals $\mathcal{S}$, we also define $R^*_{\mathcal{S}}$ as the minimum recovery threshold among all shape-preserving computation strategies, i.e.,

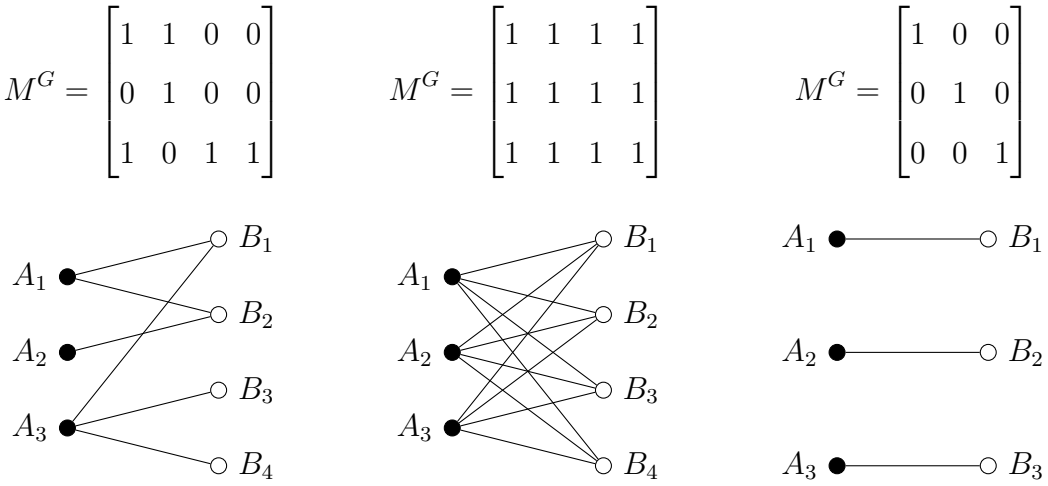
$$R^*_{\mathcal{S}} = \min_{(\mathbf{f},\mathbf{g},\mathbf{d}):\text{code is shape-preserving}} R^{(\mathbf{f},\mathbf{g},\mathbf{d})}_{\mathcal{S}}. \tag{2.8}$$

where $R^{(\mathbf{f},\mathbf{g},\mathbf{d})}_{\mathcal{S}}$ is the recovery threshold of the VCDBMM code $(\mathbf{f}, \mathbf{g}, \mathbf{d})$.

2.2 Graph Theory

In this section, we view the VCDBMM problem through graph-theoretic terms and provide necessary notation for subsequent sections. First, we observe that the desired computations $\mathcal{S}$ can be specified by using bipartite graphs. Let $G = (U \cup V, E)$ be a bipartite graph with left vertices U and right vertices V where $|U| = L_A$ and $|V| = L_B$. The left (right) vertices are given a unique label from $[L_A]$ ($[L_B]$). The edges $(i, j) \in E$ represent the desired matrix computation A_iB_j, i.e., $E = \mathcal{S}$. As such, there is a one-to-one correspondence for the computation task and the bipartite graph which allows us to use them interchangeably. For example, given a VCDBMM problem with computation goals $\mathcal{S}$ and its equivalent bipartite graph G, we re-define the optimal shape-preserving recovery threshold as $R^*_{\mathcal{G}} = R^*_{\mathcal{S}}$.

We can neatly summarize a bipartite graph using a bi-adjacency matrix $M^G \in \{0, 1\}^{|U|\times|V|}$ such that $M^G_{i,j} = 1$ implies that there is an edge between left vertex i and right vertex j. Some examples of VCDBMM problem instances using bipartite graphs are illustrated in Fig. 2.2



(a) General Example (b) Full Computation Example (c) Batch Computation Example

Figure 2.2: Examples of VCDBMM problem instances. Fig. 2.2a demonstrates a general example while Fig. 2.2b and Fig. 2.2c represent special classes of problems. Fig. 2.2b refers to the problem of getting all the pairwise computations which is solved by Polynomial codes [1]. Fig. 2.2c refers to the Batch Matrix Multiplication problem where the goal is to get all computations with matching indices for which LCC [2] and CSA codes [3] have been proposed for.

We now define terms used in graph theory to simply notation. For a left vertex x, let $N_A(x) = \{y : (x, y) \in E\}$ be the adjacent vertices to x. Similarly, for a right vertex y, let $N_B(y) = \{x : (x, y) \in E\}$ be the adjacent vertices to y. Let $d_i^A = |N_A(i)|$ ($d_j^B = |N_B(j)|$) refer to the degree of the i^{th} left vertex (j^{th} right vertex) in the graph G. Additionally, given a left vertex x, let $E_A(x) = \{(x, y) : (x, y) \in E\}$ be the set of edges adjacent to x. Similarly, given a right vertex y, let $E_B(y) = \{(x, y) : (x, y) \in E\}$ be the set of edges adjacent to y.

We define an edge partition of a bipartite graph $G = (U \cup V, E)$ as a set of sub-graphs of G where each edge of G appears in exactly one sub-graph. More precisely, $G^{\mathcal{P}}$ is an edge partition of G if $G^{\mathcal{P}} = \{G_1 = (U_1 \cup V_1, E_1), G_2 = (U_2 \cup V_2, E_2), \ldots, G_Q = (U_Q \cup V_Q, E_Q)\}$ is a collection of Q sub-graphs with $E_i \cap E_j = \emptyset,\ \forall i, j \in [Q]$ and $\cup_{q \in [Q]} E_q = E$. Given

$q \in [Q]$, let $N_A^q(i)$ be the set of adjacent vertices to the left vertex i within the sub-graph G_q. Similarly, let $N_B^q(j)$ be the set of adjacent vertices to the right vertex j within the sub-graph G_q. Additionally, let $d_{i,q}^A = |N_A^q(i)|$ and $d_{j,q}^B = |N_B^q(j)|$.

We define $\mathcal{G}_{a,b,p}$ as the random bipartite graph ensemble with a left vertices and b right vertices where each edge exists with probability p. Therefore, given a graph G with the appropriate number of vertices, $P(G \in \mathcal{G}_{a,b,p}) = p^{|E|}(1-p)^{ab-|E|}$. By the one-to-one correspondence between bipartite graphs and the VCDBMM problem, $\mathcal{G}_{a,b,p}$ can also be considered a random ensemble of VCDBMM problem instances.

2.3 Relevant Constructions

In this section, we provide a brief overview of Entangled Polynomial (EP) codes [28], Lagrange Coded Computing (LCC) codes [2], and Cross-Subspace Alignment (CSA) codes [3] that are achievable strategies for the VCDBMM problem. We consider these three schemes as a baseline for comparison to our FCSA codes.

2.3.1 Entangled Polynomial Codes

EP codes [28] are set of codes based on matrix partitioning to efficiently distribute the computation of multiplying a single pair of large matrices A and B. [1] The matrices A and B are partitioned into $n \times p$ and $p \times m$ blocks, respectively,

$$A = \begin{bmatrix} A^{1,1} & A^{1,2} & \cdots & A^{1,p} \\ A^{2,1} & A^{2,2} & \cdots & A^{2,p} \\ \vdots & \vdots & \vdots & \vdots \\ A^{n,1} & A^{n,2} & \cdots & A^{n,p} \end{bmatrix} B = \begin{bmatrix} B^{1,1} & B^{1,2} & \cdots & B^{1,m} \\ B^{2,1} & B^{2,2} & \cdots & B^{2,m} \\ \vdots & \vdots & \vdots & \vdots \\ B^{p,1} & B^{p,2} & \cdots & B^{p,m} \end{bmatrix} \tag{2.9}$$

[1] We do not mention the EP codes constructed using the bi-linear complexity approach due to the reliance of the codes on the individual properties of the input matrices such as size restrictions.

where

$$AB = \begin{bmatrix} \sum_{j=1}^{p} A^{1,j}B^{j,1} & \sum_{j=1}^{p} A^{1,j}B^{j,2} & \cdots & \sum_{j=1}^{p} A^{1,j}B^{j,m} \\ \sum_{j=1}^{p} A^{2,j}B^{j,1} & \sum_{j=1}^{p} A^{2,j}B^{j,2} & \cdots & \sum_{j=1}^{p} A^{2,j}B^{j,m} \\ \vdots & \vdots & \vdots & \vdots \\ \sum_{j=1}^{p} A^{n,j}B^{j,1} & \sum_{j=1}^{p} A^{n,j}B^{j,2} & \cdots & \sum_{j=1}^{p} A^{n,j}B^{j,m} \end{bmatrix}. \tag{2.10}$$

The partitions are encoded into the following polynomial equations

$$A(x) = \sum_{n' \in [n]} \sum_{p' \in [p]} A^{m',p'} x^{p'-1+p(m'-1)}, \tag{2.11}$$

$$B(x) = \sum_{p' \in [p]} \sum_{m' \in [m]} B^{p',n'} x^{p-p'+pm(n'-1)}. \tag{2.12}$$

The k^{th} worker receives these polynomials evaluated at x_k where $x_1, x_2, \ldots, x_K$ are all distinct values. The output of the workers is thus an evaluation of the polynomial

$$A(x)B(x) = \sum_{i=1}^{pmn+p-1} C_i x^{i-1} \tag{2.13}$$

where C_i are the resultant matrix coefficients of x^{i-1}. It can be shown that the desired blocks of Eq. (2.10) are within the coefficients $C_1, C_2, \ldots, C_{pmn+p-1}$. Additionally, these coefficients can be retrieved from any $pmn + p - 1$ evaluations of $A(x)B(x)$ by inverting a Vandermonde matrix. Thus, the recovery threshold of such a scheme is $R = pmn + p - 1$.

EP codes are a generalization of Polynomial codes [1] ($p = 1$) and MatDot codes [21] ($n = m = 1$). Polynomial codes are of particular interest to us because they can be used to solve the VCDBMM problem under the shape-preserving constraint. Given $\mathcal{A}$ and $\mathcal{B}$, we construct two matrices $\mathbf{A}$ and $\mathbf{B}$ as

$$\mathbf{A} = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_{L_A} \end{bmatrix}, \mathbf{B} = \begin{bmatrix} B_1 & B_2 & \cdots & B_{L_B} \end{bmatrix} \tag{2.14}$$

where

$$\mathbf{AB} = \begin{bmatrix} A_1B_1 & A_1B_2 & \cdots & A_1B_{L_B} \\ A_2B_1 & A_2B_2 & \cdots & A_2B_{L_B} \\ \vdots & \vdots & \vdots & \vdots \\ A_{L_A}B_1 & A_{L_A}B_2 & \cdots & A_{L_A}B_{L_B} \end{bmatrix}. \tag{2.15}$$

By treating each matrix as a partition in $\mathbf{A}$ and $\mathbf{B}$, Polynomial codes can solve the VCDBMM problem by calculating all pairs of A_i and B_j which would allow for a recovery threshold of L_AL_B. While such a computation strategy would clearly not be optimal for a general VCDBMM problem, it offers a good upper bound on achievable schemes and, in fact, performs better than some of the other codes for highly dense graphs. Additionally, Polynomial codes are shape-preserving if the matrices are not further partitioned.

2.3.2 Lagrange Coded Computing Codes

Lagrange Coded Computing (LCC) codes [2] are a class of codes designed for distributed batch multivariate polynomial evaluation which includes distributed batch matrix multiplication. The name of LLC codes comes from their use of the Lagrange interpolation polynomial to encode data.

Consider a multivariate polynomial $\Gamma(\cdot)$ of degree N and suppose we want to get the evaluations of $\Gamma(\cdot)$ for a batch of data points $z_1, z_2, \cdots, z_L$. LCC codes encode the batch using the Lagrange interpolation polynomial

$$\widetilde{Z}(x) = \sum_{i\in[L]} z_i \prod_{j\in[L]\setminus\{i\}} \frac{(x-\zeta_j)}{(\zeta_i-\zeta_j)} \tag{2.16}$$

where $\zeta_1, \zeta_2, \ldots, \zeta_L$ are distinct constants in $\mathbb{F}$. Given distinct $x_1, x_2, \ldots, x_K$, the k^{th} worker receives $\widetilde{Z}(x_k)$, evaluates $\Gamma(\widetilde{Z}(x_k))$, and returns the result to the fusion node. Note that $\Gamma(\widetilde{Z}(x))$ is a polynomial of degree at most $N(L-1)$ and thus can be fully interpolated from $N(L-1)+1$ evaluations. Once the fusion node interpolates $\Gamma(\widetilde{Z}(x))$, it can acquire $\Gamma(z_i)$ by evaluating $\Gamma(\widetilde{Z}(x))$ at ζ_i, i.e., $\Gamma(\widetilde{Z}(\zeta_i)) = \Gamma(z_i)$.

Note that this framework allows for batch matrix multiplication by defining $\Gamma(A, B) = AB$ and letting $Z_i = (A_i, B_i)$ which makes it a bi-linear operation ($N = 2$). Thus, the recovery threshold is $2(L-1)+1 = 2L-1$. We observe that this is a special case of the VCDBMM problem when $L_A = L_B = L$, $\mathcal{A} = \{A_1, A_2, \ldots, A_L\}$, $\mathcal{B} = \{B_1, B_2, \ldots, B_L\}$, and $\mathcal{S} = \{(i,i) : i \in [L]\}$. This special case can be seen in Fig. 2.2c.

LCC codes can be used to solve the general VCDBMM problem by applying a simple trick. Given a bipartite graph G, assign an arbitrary order to the edges from 1 to $|E|$. For the i^{th} edge in G, we create a data point (A_i, B_j) that captures the desired computation. Thus, there are $|E|$ data points in the batch which results in a recovery threshold of $2|E|-1$. Similar to EP codes, we use LCC codes as a baseline to compare FCSA codes to.

2.3.3 Cross-Subspace Alignment Codes

We now describe a class of codes that are a precursor to FCSA codes: Cross-Subspace Alignment (CSA) codes. We first mention an important lemma that we rely on heavily throughout this thesis. This lemma is a standard result for Confluent Cauchy-Vandermonde matrices [37] and was the building block for CSA codes [3, 38]. While similar proofs are shown in [37, 38], we provide our own brief proof for completeness.

Lemma 1. *Let $f_1, f_2, \cdots, f_N, x_1, x_2, \cdots, x_K$ be $N+K$ distinct elements of $\mathbb{F}$, with $|\mathbb{F}| \geq N+K$. Each element f_i has an associated multiplicity of u_i where $u_1 + u_2 + \cdots + u_N = M$. Let $M+1 \leq K$. The following $K \times K$ Confluent Cauchy-Vandermonde matrix is invertible over $\mathbb{F}$:*

$$\mathbf{V}_{N,K,\{u_i\}_{i=1}^N} \triangleq \begin{bmatrix} \frac{1}{(x_1-f_1)^{u_1}} & \cdots & \frac{1}{x_1-f_1} & \cdots & \frac{1}{(x_1-f_N)^{u_N}} & \cdots & \frac{1}{x_1-f_N} & 1 & x_1 & \cdots & x_1^{K-M-1} \\ \frac{1}{(x_2-f_1)^{u_1}} & \cdots & \frac{1}{x_2-f_1} & \cdots & \frac{1}{(x_2-f_N)^{u_N}} & \cdots & \frac{1}{x_2-f_N} & 1 & x_2 & \cdots & x_2^{K-M-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{1}{(x_K-f_1)^{u_1}} & \cdots & \frac{1}{x_K-f_1} & \cdots & \frac{1}{(x_K-f_N)^{u_N}} & \cdots & \frac{1}{x_1-f_N} & 1 & x_K & \cdots & x_K^{K-M-1} \end{bmatrix} \tag{2.17}$$

Proof. See Appendix A.1. □

Intuitively, we can understand Lemma 1 in the following way. Consider the function $h(x)$ where

$$h(x) = \sum_{i=1}^{N} \sum_{j=1}^{u_i} \frac{c_i}{(f_i - x)^j} + \sum_{i=M+1}^{K} c_i x^{i-M-1} \tag{2.18}$$

and c_i are some constants. Lemma 1 states that $h(x)$ can be uniquely interpolated in $\mathbb{F}$ using the K evaluations $h(x_i)$ as long as all elements of $\{x_i : i \in [K]\}$ and $\{f_i : i \in [N]\}$ are distinct and the field $\mathbb{F}$ is sufficiently large. Thus, the coefficients c_i can be extracted.

The main idea behind CSA codes is that the desired computations are encoded in the rational terms $\frac{1}{(f_i - x_k)^r}$ while all undesired computations (interference terms) are encoded in the polynomial terms x_k^r. We now provide an example of CSA codes. Similar to LCC codes, CSA codes are designed for the Batch Matrix Multiplication problem. As such, let $\mathcal{A} = \{A_1, A_2\}$, $\mathcal{B} = \{B_1, B_2\}$, and $\mathcal{S} = \{(1,1), (2,2)\}$, i.e., the goal is to compute $A_1 B_1$ and $A_2 B_2$. Let f_1 and f_2 be unique elements in $\mathbb{F}$. Consider the following encoding polynomials

$$\begin{aligned} A(x) &= (x - f_1)(x - f_2)\left(\frac{A_1}{x - f_1} + \frac{A_2}{x - f_2}\right) \qquad (2.19) \\ &= (x - f_2)A_1 + (x - f_1)A_2 \\ B(x) &= \frac{B_1}{x - f_1} + \frac{B_2}{x - f_2}. \qquad (2.20) \end{aligned}$$

The transmitted matrices $\widetilde{A}_k$ and $\widetilde{B}_k$ are different evaluations of the previous polynomials, i.e., $\widetilde{A}_k = A(x_k)$ and $\widetilde{B}_k = B(x_k)$ for some x_k distinct from f_1 and f_2. As such, the result from a worker is an evaluation of the following polynomial

$$A(x)B(x) = \frac{A_1 B_1}{x - f_1} + \frac{A_2 B_2}{x - f_1} + I \tag{2.21}$$

where I is some arbitrary interference term. By Lemma 1, $A(x)B(x)$ can be interpolated using 3 evaluations and the recovery threshold of CSA codes is thus 3. In general, given L

Code	Recovery Threshold
Polynomial	$L_A L_B$
LCC	$2\|E\| - 1$
CSA	$2\|E\| - 1$
FCSA Type 1	$2\|E\| - \min_{i\in[L_A]} d_i^A - \min_{i\in[L_B]} d_i^B + 1$
FCSA Type 2	$\ll 2\|E\| - \min_{i\in[L_A]} d_i^A - \min_{i\in[L_B]} d_i^B + 1$

Table 2.1: Recovery thresholds for shape-preserving VCDBMM codes.

pairs of matrices, CSA codes have a recovery threshold of $2L - 1$. [2]

Since CSA codes solve the Batch Matrix Multiplication problem, we use a similar technique as for LCC codes and duplicate the matrices to ensure that the bipartite graph G has only isolated edges. As such, CSA codes provide the same recovery threshold as LCC codes which is $2|E| - 1$. Despite their similar recovery threshold to LCC codes, CSA codes provide a lot of flexibility in code design when it comes to communication and computation complexity when the shape-preserving constraint is relaxed. Since FCSA codes are based on CSA codes, they offer the same flexibility but require more nuanced analysis to better take advantage of this flexibility as will be shown in Chapter 5. In the next chapter, we present the first variant of FCSA codes that provide a better recovery threshold than Polynomial, LLC, and CSA codes for the VCDBMM problem. Before concluding this chapter, we present a table of recovery thresholds for FCSA codes and the relevant constructions, under the shape-preserving constraint, in Table 2.1.

[2]CSA codes can have a smaller recovery threshold if the matrices are allowed to be grouped up and each group is individually encoded using MDS codes. The recovery threshold presented only considers when there is one group. We do this intentionally because we cannot give a definitive recovery threshold when grouping is allowed for the VCDBMM problem. We address this issue in Chapter 5.

CHAPTER 3

Flexible Cross-Subspace Alignment Type 1 codes

In this chapter, we present the first variant of our codes that provide an achievable scheme to solve the VCDBMM problem. We name these codes as Flexible Cross-Subspace Alignment Type 1 (FCSA-T1) codes. In the following theorem, we present our coding construction and prove that FCSA-T1 codes are within a factor of two optimal among all shape preserving codes.

Theorem 1. *Let $G = (U \cup V, E)$ be a VCDBMM bipartite graph. Assuming that $|\mathbb{F}| > |E|+K$, FCSA Type 1 codes achieve a recovery threshold of*

$$R_{FCSA\text{-}T1} = 2|E| - \min_{i\in[L_A]}(d_i^A) - \min_{i\in[L_B]}(d_i^B) + 1 \tag{3.1}$$

while being shape-preserving.

Conversely, the recovery threshold of this code is within a multiplicative factor of two from the optimal recovery threshold $R_{\mathcal{S}}^*$ among all shape-preserving VCDBMM codes, i.e.,

$$R_{\mathcal{S}}^* \leq R_{FCSA\text{-}T1} < 2R_{\mathcal{S}}^* \tag{3.2}$$

3.1 Achievability of Theorem 1

In this section, we prove the achievability of Theorem 1 by constructing FCSA-T1 codes that attains the recovery threshold of $R_{\text{FCSA-T1}}$.

Let $G = (U \cup V, E)$ be a graph representing the VCDBMM problem. Assume that every vertex has at least one edge connected to it, i.e., no isolated vertices, otherwise we prune

the graph of these vertices without affecting the computation goals. Additionally, assume $2|E| - \min_{i\in[L_A]}(d_i^A) - \min_{i\in[L_B]}(d_i^B) + 1 \leq L_A L_B$ otherwise we can use the Polynomial codes to achieve a recovery threshold of $L_A L_B$ as shown in section 2.3.1.

For each edge $e \in E$, we associate a distinct element f_e from $\mathbb{F}$. We select K additional distinct elements from $\mathbb{F}$ and represent them as $x_1, x_2, \ldots, x_K$. Recall that $E_A(i)$ ($E_B(j)$) is the set of edges adjacent to the left vertex i (right vertex j). Let us define

$$a_i(x) = \prod_{e\in E_A(i)} (x - f_e), \tag{3.3}$$

$$b_j(x) = \prod_{e\in E_B(j)} (x - f_e). \tag{3.4}$$

Note that the degrees of the polynomials $a_i(x)$ and $b_j(x)$ are $\deg(a_i(x)) = d_i^A$ and $\deg(b_j(x)) = d_j^B$. Additionally, we observe that $a_i(x)$ and $b_j(x)$ can share at most one root which is $f_{(i,j)}$. We also define

$$\Theta^{\mathrm{T1}}(x) = \prod_{e\in E}(x - f_e). \tag{3.5}$$

Source $\mathbb{A}$ uses the following polynomial to encode the matrices

$$A^{\mathrm{T1}}(x) = \Theta(x) \sum_{i=1}^{L_A} \frac{1}{a_i(x)} A_i.$$

Similarly, Source $\mathbb{B}$ uses

$$B^{\mathrm{T1}}(x) = \sum_{j=1}^{L_B} \frac{1}{b_j(x)} B_j.$$

For the k^{th} worker node, the source nodes transmit the evaluations of the encoding polynomial at x_k, i.e $\widetilde{A}_k = A^{\mathrm{T1}}(x_k)$ and $\widetilde{B}_k = B^{\mathrm{T1}}(x_k)$. We now drop the subscript on x_k for ease of notation.

The resulting computation from a worker is thus an evaluation of the function

$$A^{\mathrm{T1}}(x)^{\mathrm{T1}}B(x) = \Theta^{\mathrm{T1}}(x)(\sum_{i=1}^{L_A}\frac{1}{a_i(x)}A_i)(\sum_{j=1}^{L_B}\frac{1}{b_j(x)}B_j) \tag{3.6}$$

$$= \sum_{i=1}^{L_A}\sum_{j=1}^{L_B}\frac{\Theta^{\mathrm{T1}}(x)}{a_i(x)b_j(x)}A_iB_j \tag{3.7}$$

$$= \sum_{(i,j)\notin E}\prod_{e\in\widetilde{E}_{i,j}}(x-f_e)A_iB_j + \sum_{(i,j)\in E}\prod_{e\in\widetilde{E}_{i,j}}\frac{(x-f_e)}{(x-f_{(i,j)})}A_iB_j \tag{3.8}$$

where $\widetilde{E}_{i,j} = \{e : e \in E, e \notin E_A(i), e \notin E_B(j)\}$. The first term in (3.8) corresponds to the undesired computations, i.e., $(i,j) \notin E$, and the second term in (3.8) corresponds to the desired computations, i.e $(i,j) \in E$.

We observe that the first term in (3.8) is a polynomial in x with degree $\max_{(i,j)\notin E}|\widetilde{E}_{i,j}|$. It is clear from context that for $(i,j) \notin E$ then $|\widetilde{E}_{i,j}| = |E| - (d_i^A + d_j^B)$. Therefore, the first term creates a polynomial of degree $|E| - \min_{(i,j)\notin E}(d_i^A + d_j^B)$.

Now, let us consider the second term in (3.8). By the rules of partial fraction decomposition, we expand the product

$$\prod_{e\in\widetilde{E}_{i,j}}\frac{(x-f_e)}{(x-f_{i,j})}A_iB_j \tag{3.9}$$

into a weighted sum

$$(\frac{\zeta_{-1}}{(x-f_{i,j})} + \zeta_0 + \zeta_1 x + \cdots + \zeta_{|\widetilde{E}_{i,j}|-1}x^{|\widetilde{E}_{i,j}|-1})A_iB_j \tag{3.10}$$

for some constants $\{\zeta_k\}$. By the polynomial remainder theorem, we have that $\zeta_{-1} \neq 0$ since $\prod_{e\in\widetilde{E}_{i,j}}(f_{(i,j)} - f_e) \neq 0$. Given $(i,j) \in E$, it is obvious that $|\widetilde{E}_{i,j}| = |E| - (d_i^A + d_j^B) + 1$. Therefore, the expression in Eq. (3.9) produces a single rational term and a polynomial of degree $|E| - (d_i^A + d_j^B)$. Thus, the second term in Eq. (3.8) produces a polynomial of degree $|E| - \min_{(i,j)\in E}(d_i^A + d_j^B)$.

Combining the two expressions in (3.8) results in $|E|$ rational terms and a polynomial with degree $|E| - \min_{(i,j)\in[L_A,L_B]}(d_i^A + d_j^B) = |E| - \min_{i\in[L_A]}(d_i^A) - \min_{i\in[L_B]}(d_i^B)$. Let $T =$

$|E| - \min_{i\in[L_A]}(d_i^A) - \min_{i\in[L_B]}(d_i^B)$. Thus, we write (3.8) as

$$A^{\mathrm{T1}}(x)B^{\mathrm{T1}}(x) = \sum_{(i,j)\in E} \frac{\zeta_{i,j}}{(x - f_{(i,j)})} A_i B_j + \sum_{r=0}^{T} I_r x^r \tag{3.11}$$

for some non-zero constants $\{\zeta_{i,j} : (i,j) \in E\}$ and interference terms $\{I_r : r \in \{0, 1, \ldots, T\}\}$.

By Lemma 1, $A^{\mathrm{T1}}(x)B^{\mathrm{T1}}(x)$ can be interpolated from $2|E| - \min_{i\in[L_A]}(d_i^A) - \min_{i\in[L_B]}(d_i^B) + 1$ evaluations. As such, we acquire the scaled versions of $\{A_i B_j : (i,j) \in \mathcal{S}\}$ which can be easily re-scaled to extract the desired computation. Thus, we prove the achievability of the recovery threshold $R_{\text{FCSA-T1}}$.

3.2 Complexity Analysis of FCSA-T1 codes

Computational and Communication Complexity: Since FCSA-T1 codes are shape-preserving codes, we quickly ascertain that the upload costs are $U_A = \alpha\beta$ and $U_B = \beta\gamma$ and the download cost is $D_C = \alpha\gamma$. Additionally, the computational complexity at each worker is $\mathcal{O}(\alpha\beta\gamma)$.

Encoding and Decoding Complexity: Encoding the matrices $\widetilde{A}_k$ for $k \in [K]$ involves scaling L_A matrices with $\alpha\beta$ elements each and adding them up. For each matrix A_k, this requires a complexity of $\mathcal{O}(L_A\alpha\beta)$. Thus, the overall encoding complexity at source $\mathcal{A}$ is $\mathcal{O}(KL_A\alpha\beta)$. Similarly, the overall encoding complexity at source $\mathcal{B}$ is $\mathcal{O}(KL_B\beta\gamma)$

The decoding complexity involves inverting a $R_{\text{FCSA-T1}} \times R_{\text{FCSA-T1}}$ confluent Cauchy-Vandermonde matrix for which the best known algorithm is $\widetilde{\mathcal{O}}(R_{\text{FCSA-T1}} \log^2 R_{\text{FCSA-T1}})$ [39,40]. This inversion is performed for all $\alpha\gamma$ symbols in the output. Since re-scaling the matrices takes negligible time, the total decoding complexity is $\widetilde{\mathcal{O}}(\alpha\gamma \cdot R_{\text{FCSA-T1}} \log^2 R_{\text{FCSA-T1}})$.

3.3 Converse Lower Bound for VCDBMM

Now, we prove the converse lower bound demonstrating that FCSA-T1 codes are within a factor of two optimal among shape-preserving VCDBMM codes. To prove this, we first note that $R_{\text{FCSA-T1}} \leq 2|E|$. As such, it is sufficient to show that $|E| \leq R^*_G$ to guarantee $R_{\text{FCSA-T1}} \leq 2R^*_G$. We prove this in the following Lemma.

Lemma 2. *Given a VCDBMM bipartite graph $G = (U \cup V, E)$, $|E| \leq R^*_G$.*

Proof. First, we note that the problem definition specifies that the computation strategy has to work for all $\mathcal{A}$ and $\mathcal{B}$, as long as the field $\mathbb{F}$ is sufficiently large. We can thus prove the lower bound by constraining $\mathcal{A}$ and $\mathcal{B}$ to a certain class of matrices and providing a lower bound for this class.

Assume that elements of $\mathcal{A}$ and $\mathcal{B}$ are from a finite field of size q, i.e., $\mathbb{F}_q$. Consider a matrix $\mathbf{B} \in \mathbb{F}_q^{\beta \times L_B \gamma}$ that is constructed by horizontally concatenating all the matrices in $\mathcal{B}$, i.e., $\mathbf{B} = \begin{bmatrix} B_1 & B_2 & \cdots & B_{L_B} \end{bmatrix}$. We assume that the matrices in $\mathcal{B}$ are chosen such that $\mathbf{B}$ is a tall matrix (i.e., $\beta > L_B\gamma$) and that $\mathbf{B}$ is full rank. Let $\mathcal{B}$ be fixed.

Let all the matrices in $\mathcal{A}$ be uniformly sampled from $\mathbb{F}_q^{\alpha \times \beta}$. As such, we treat the symbols in A_iB_j as random variables that are uniformly distributed on $\mathbb{F}_q^{\alpha \times \gamma}$. For a fixed $i \in [L_A]$, the computation A_iB_j is a sub-matrix of $A_i\mathbf{B}$. By the full rank property of $\mathbf{B}$, A_iB_j are independent random variables for all $i \in [L_A]$ and $j \in [L_B]$. Therefore, the entropy of the desired computations is $H(\{A_iB_j : (i,j) \in \mathcal{S}\}) = |\mathcal{S}|\alpha\gamma \log_2 q = |E|\alpha\gamma \log_2 q$. To reconstruct this set of random variables, the fusion node needs at least $|E|\alpha\gamma \log_2 q$ bits. By the shape-preserving constraint, each worker only provides $\alpha\gamma \log_2 q$ bits to the fusion node. Using a cut-set bound argument, the fusion node needs at least $|E|$ responses from the workers to reconstruct $\{A_iB_j : (i,j) \in \mathcal{S}\}$ which completes the proof. □

Remark. *We note that the proof of Lemma 2 relied on the assumption that the input matrices come from a finite field to provide a general lower bound across all input matrices. In general,*

constraining the class of input matrices provides additional information for which a better code can be constructed. Conversely, this also means that the bound found in Lemma 2 need not necessarily be tight for all input fields. For example, authors in [22] determined two lower bounds for EP codes, one for general fields and one for finite fields. The bound for finite fields was much lower than for general input fields which was shown to be reachable by [30] where authors use the properties of conjugates in finite fields to reduce the recovery threshold. As such, we speculate that the bound for VCDBMM can be further refined by constraining parts of the system and consider it a interesting future research topic.

3.4 FCSA-T1 Performance on $\mathcal{G}_{L_A,L_B,p}$

We are proposing FCSA-T1 codes as a general solution for a wide-variety of VCDBMM problem instances. As such, we wish to demonstrate the average performance of FCSA-T1 codes on the ensemble $\mathcal{G}_{L_A,L_B,p}$. We primarily focus on the average improvement offered by FCSA-T1 codes over naive applications of previous solutions, i.e., CSA codes. Therefore, we define the average improvement as $\mathbb{E}[R_{Naive} - R_{\text{FCSA-T1}}]$ where R_{Naive} is the recovery threshold of either LCC or CSA codes, i.e., $\mathbb{E}[R_{Naive} = 2|E| - 1$. In the following theorem, we demonstrate that the average improvement has an asymptotic growth rate that is at least linear in the number of vertices.

Lemma 3. *For a fixed $0 < p < 1$, the average improvement for graphs in the ensemble $\mathcal{G}_{L_A,L_B,p}$ has a linear growth rate in terms of the number of vertices. Specifically,*

$$\mathbb{E}[R_{Naive} - R_{FCSA\text{-}T1}] = \mathbb{E}[\min_{i\in[L_A]} d_i^A + \min_{i\in[L_B]} d_i^B] = \Omega(L_A + L_B). \tag{3.12}$$

Proof. See Appendix A.2. □

To confirm Lemma 3, we provide simulations for the average improvement. In Fig. 3.1, we simulate the average improvement across varying L_A for different values of p with $L_B = 5$. For all values of p, we notice a clear linear trend as stated by Lemma 3. We can also see

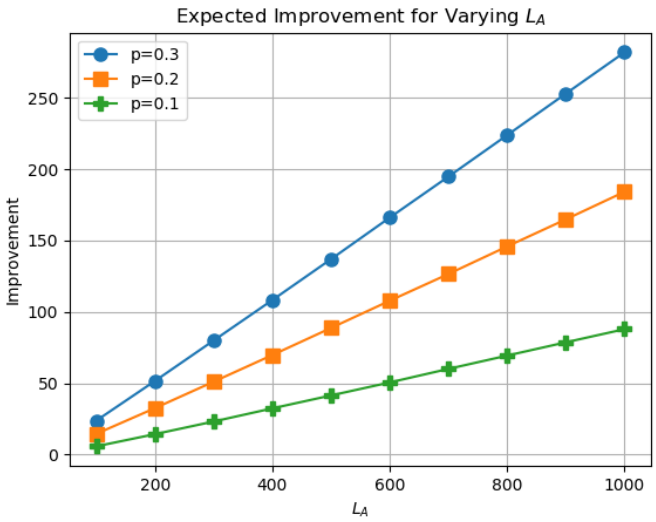


Figure 3.1: Shows the average improvement for the recovery threshold using FCSA-T1 codes across varying L_A, fixed values for p, and $L_B = 5$.

that the improvement gets better for larger values of p. Both of these trends indicate that FCSA-T1 codes provide better improvement for denser graphs.

We also simulate the average improvement across various graph ensembles with the same average number of edges. By keeping the average number of edges the same, all the ensembles have the same lower bound for the optimal recovery threshold guaranteed by Lemma 2. The plot in Fig. 3.2 shows the average improvement for various (L_A, L_B) pairs where $L_A L_B$ is kept constant. From the plot, we note that the best improvement happens for the pair $(1000, 5)$ and the worst improvement happens for $(100, 50)$. We speculate that this trend occurs because the pair $(1000, 5)$ allows for a large concentration of edges on right vertices which indicate that these matrices are re-used in many computations. FCSA-T1 codes provide a large improvement due to this redundancy in computation. On the other hand, the $(100, 50)$ pair allows for less concentration of edges which limits the improvement by FCSA-T1 codes. This may indicate that either the naive coding solutions may already be close to optimal for these sparse computation graphs or that FCSA-T1 codes do not take full-advantage of the

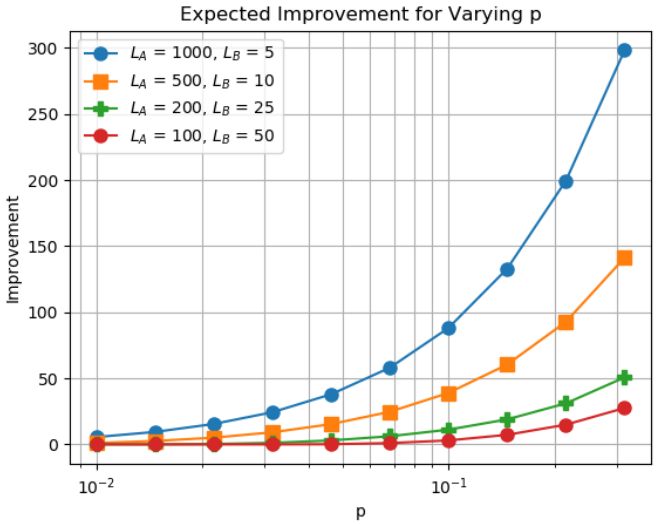


Figure 3.2: Shows the average improvement for the recovery threshold using FCSA-T1 codes across varying p and fixed pairs of (L_A, L_B).

redundancy in computations. While the first statement may be true for extremely sparse graphs, in the next chapter we demonstrate how to enhance FCSA-T1 codes to further reduce the recovery threshold.

CHAPTER 4

Flexible Cross-Subspace Alignment Type 2 codes: Improved Recovery Threshold by Optimization

In this chapter, we demonstrate how to improve the recovery threshold of the FCSA-T1 codes at the cost of increased encoding and decoding complexity. We designate these new codes as FCSA Type 2 (FCSA-T2) codes. To create FCSA-T2 codes, we propose a general class of shape-preserving VCDBMM codes and use a mixed-integer program in order to select a code with the minimum recovery threshold.

4.1 Motivating Example

Consider the following bi-adjacency matrix of a bipartite graph G:

$$M^G = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

Assume that $f_{1,1}, f_{1,2}, f_{2,1}$ are unique elements from $\mathbb{F}$. We construct FCSA-T1 codes by the following encoding polynomials

$$\Theta_1^{\mathrm{T1}}(x) = (x - f_{(1,1)})(x - f_{(1,2)})(x - f_{(2,1)}), \tag{4.1}$$

$$A_1^{\mathrm{T1}}(x) = \Theta_1^{\mathrm{T1}}(x)\left(\frac{A_1}{(x - f_{(1,1)})(x - f_{(1,2)})} + \frac{A_2}{(x - f_{(2,1)})}\right), \tag{4.2}$$

$$B_1^{\mathrm{T1}}(x) = \frac{B_1}{(x - f_{(1,1)})} + \frac{B_2}{(x - f_{(1,2)})(x - f_{(2,1)})} \tag{4.3}$$

where

$$A_1^{\mathrm{T1}}(x)B_1^{\mathrm{T1}}(x) = \frac{\zeta_{(1,1)}}{(x-f_{(1,1)})}A_1B_1 + \frac{\zeta_{(1,2)}}{(x-f_{(1,2)})}A_1B_2 + \frac{\zeta_{(2,1)}}{(x-f_{(2,1)})}A_2B_1 + I_0 + I_1x \quad (4.4)$$

for some non-zero constants $\zeta_{(1,1)}, \zeta_{(1,2)}, \zeta_{(2,1)}$ and interference terms I_0,I_1. Intuitively, we see that FCSA-T1 codes aim to align each desired computation to its own unique linear sub-space and that everything else is thrown into the interference terms. Such an approach can be considered edge-centric since each edge of the VCDBMM problem is assigned its own sub-space and the sub-spaces are not allowed to interfere with each other. While this allows for quick retrieval of the desired computations after interpolating the polynomial, it is not the best approach to improve the recovery threshold.

Now, let f_1, f_2 be unique elements from $\mathbb{F}$ which we associate to A_1 and A_2. Consider the following encoding polynomials

$$\Theta_1^{\mathrm{T2}}(x) = (x-f_1)^2(x-f_2), \quad (4.5)$$

$$A_1^{\mathrm{T2}}(x) = \Theta_1^{\mathrm{T2}}(x)(\frac{A_1}{(x-f_1)^2} + \frac{A_2}{(x-f_2)}), \quad (4.6)$$

$$B_1^{\mathrm{T2}}(x) = \frac{B_1}{(x-f_1)^2} + \frac{B_2}{(x-f_2)(x-f_1)}. \quad (4.7)$$

Now, consider

$$A_1^{\mathrm{T2}}(x)B_1^{\mathrm{T2}}(x) = \frac{(x-f_2)}{(x-f_1)^2}A_1B_1 + \frac{1}{(x-f_1)}A_1B_2 + \frac{(x-f_1)}{(x-f_2)}A_2B_2 + A_2B_1 \quad (4.8)$$

which can be expanded by partial fraction decomposition into

$$A_1^{\mathrm{T2}}(x)B_1^{\mathrm{T2}}(x) = \frac{\zeta_0}{(x-f_1)^2}A_1B_1 + \frac{1}{(x-f_1)}(\zeta_1A_1B_1 + \zeta_2A_1B_2) + \frac{\zeta_3}{(x-f_2)}A_2B_2 + I_0 \quad (4.9)$$

for non-zero constants $\zeta_0, \zeta_2, \zeta_3$, constant ζ_1, and interference term I_0. Since $\zeta_0, \zeta_2, \zeta_3$ are non-zero, we easily extract A_1B_1, A_1B_2,and A_2B_2 after interpolating the function which only requires 4 evaluations. In this example, we designated a sub-space for each left vertex such that all computations associated with the vertex have their own sub-space. For our codes, the sub-spaces are the powers of the rational terms. In the above example, the first sub-space

was for the rational terms related to $\frac{1}{(x-f_1)^2}$ and $\frac{1}{(x-f_1)}$ and the second sub-space was for the rational term $\frac{1}{(x-f_2)}$. This code is accomplished by assigning an element to each left vertex and determining the power associated to each edge. We consider this as a vertex-centric approach which assigns the computations associated to a vertex into one subspace instead of just one edge. This has been an example of FCSA-T2 codes.

To demonstrate how the power assignment can affect the recovery threshold, consider the following encoding polynomials

$$\Theta_2^{\mathrm{T2}}(x) = (x-f_1)^2(x-f_2), \tag{4.10}$$

$$A_2^{\mathrm{T2}}(x) = \Theta_2^{\mathrm{T2}}(x)(\frac{A_1}{(x-f_1)^2} + \frac{A_2}{(x-f_2)}), \tag{4.11}$$

$$B_2^{\mathrm{T2}}(x) = \frac{B_1}{(x-f_1)} + \frac{B_2}{(x-f_2)(x-f_1)^2}. \tag{4.12}$$

Note that only difference is that the denominators in $B_2^{\mathrm{T2}}(x)$ are different. Now, consider

$$A_2^{\mathrm{T2}}(x)B_2^{\mathrm{T2}}(x) = \frac{(x-f_2)}{(x-f_1)}A_1B_1 + \frac{1}{(x-f_1)^2}A_1B_2 + \frac{1}{(x-f_2)}A_2B_2 + (x-f_1)A_2B_1 \tag{4.13}$$

which can be expanded by partial fraction decomposition into

$$A_2^{\mathrm{T2}}(x)B_2^{\mathrm{T2}}(x) = \frac{\zeta_0}{(x-f_1)^2}A_1B_1 + \frac{1}{(x-f_1)}(\zeta_1 A_1B_1 + \zeta_2 A_1B_2) + \frac{\zeta_3}{(x-f_2)}A_2B_2 + I_0 + I_1x \tag{4.14}$$

for non-zero constants $\zeta_0, \zeta_2, \zeta_3$, constant ζ_1, and interference terms I_0, I_1. Again, we extract A_1B_1, A_1B_2, and A_2B_2 from the function by a similar argument as in the last example. But, this function requires 5 evaluations to be fully interpolated. As such, the power assignment greatly affects the recovery threshold of the code.

In the next section, we formally define FCSA-T2 codes and demonstrate how to construct the codes using the power assignments.

4.2 Construction of FCSA-T2 Codes

First, we define the left power assignment matrix which we use to construct FCSA-T2 codes.

Definition 1. *P is a left power assignment matrix for a graph G if it satisfies the following conditions:*

- *Given a tuple (i,j), $M^G_{i,j} = 0 \implies P_{i,j} = 0$.*
- *Given a tuple (i,j), $M^G_{i,j} = 1 \implies P_{i,j} \in \{1, 2, \ldots, d^A_i\}$.*
- *Given a tuple (i,j,k) where $j \neq k$, if $P_{i,j} \neq 0$ and $P_{i,k} \neq 0$ then $P_{i,j} \neq P_{i,k}$ for $j \neq k$.*

Note that we can similarly define a right power assignment matrix if we flip the graph such that the right vertices becomes left vertices and vice-versa. As such, we focus purely on constructing codes using left power assignment matrices. Thus, whenever we mention power assignment matrices, we are referring to left power assignments matrices.

Given a power assignment matrix P, we show how to construct a VCDBMM code with a well-defined recovery threshold. This is characterized by the following theorem.

Theorem 2. *Let $G = (U \cup V, E)$ be a VCDBMM bipartite graph and let P be a power assignment matrix. Assuming that $|\mathbb{F}| > L_A + K$, FCSA Type 2 codes achieve a recovery threshold of*

$$R_{FCSA\text{-}T2} = 2|E| - \min_{i \in [L_A]}(d^A_i) - \min_{j \in [L_B]}(\sum_{i=1}^{L_A} P_{i,j}) + 1 \tag{4.15}$$

while being shape-preserving.

Proof. To prove this theorem, we now construct FCSA-T2 codes. The construction follows a similar logical flow as for FCSA-T1 codes with some key differences.

For each left vertex $i \in [L_A]$, we associate a distinct element f_i from $\mathbb{F}$. We select K additional distinct elements from $\mathbb{F}$ and represent them as $x_1, x_2, \ldots, x_K$. Recall that $N_A(i)$ ($N_B(j)$) is the set of vertices adjacent to the left vertex i (right vertex j).

Given an edge $(i,j) \in E$, we define

$$\Delta_{i,j}(x) = \prod_{r \notin N_B(j)} (x + (f_i - f_r))^{d^A_r} \prod_{r \in N_B(j) \setminus \{i\}} (x + (f_i - f_r))^{d^A_r - P_{r,j}} = \sum_{l=0}^{D_{i,j}} \zeta_{i,j,l} x^l \tag{4.16}$$

where $D_{i,j}$ is the degree of $\Delta_{i,j}(x)$ and $\zeta_{i,j,l}, l \in \{0, 1, \ldots, D_{i,j}\}$ are the coefficients of $\Delta_{i,j}(x)$. We observe that

$$D_{i,j} = \sum_{r \notin N_B(j)} d_r^A + \sum_{r \in N_B(j) \setminus \{i\}} d_r^A - \sum_{r \in N_B(j) \setminus \{i\}} P_{r,j} \tag{4.17}$$

$$= \sum_{r \in [L_A]} d_r^A - d_i^A - \sum_{r \in N_B(j) \setminus \{i\}} P_{r,j} \tag{4.18}$$

$$= |E| - d_i^A - \sum_{r=1}^{L_A} P_{r,j} + P_{i,j} \tag{4.19}$$

where Eq. (4.19) uses the fact that $P_{i,j} = 0$ for $(i,j) \notin E$.

Let us define

$$a_i(x) = (x - f_i)^{d_i^A}, \tag{4.20}$$

$$b_j(x) = \prod_{i \in N_B(j)} (x - f_i)^{P_{i,j}} \tag{4.21}$$

Note that the degrees of the polynomials of $a_i(x)$ and $b_j(x)$ are $\deg(a_i(x)) = d_i^A$ and $\deg(b_j(x)) = \sum_{i=1}^{L_A} P_{i,j}$ since $P_{i,j} = 0$ for $(i,j) \notin E$. Additionally, note that $a_i(x)$ and $b_j(x)$ only share a root if $(i,j) \in E$ and that root is f_i.

We define

$$\Theta^{\mathrm{T2}}(x) = \prod_{i \in [L_A]} (x - f_i)^{d_i^A} = \prod_{i \in [L_A]} a_i(x). \tag{4.22}$$

Note that the polynomial degree of $\Theta(x)$ is $\sum_{i \in [L_A]} d_i^A = |E|$.

Source $\mathbb{A}$ uses the following polynomial to encode the matrices

$$A^{\mathrm{T2}}(x) = \Theta^{\mathrm{T2}}(x) \sum_{i=1}^{L_A} \frac{1}{a_i(x)} A_i.$$

Similarly, Source $\mathbb{B}$ uses

$$B^{\mathrm{T2}}(x) = \sum_{j=1}^{L_B} \frac{1}{b_j(x)} B_j.$$

For the k^{th} worker node, the source nodes transmit the evaluations of the encoding polynomial at x_k, i.e $A^{\mathrm{T2}}(x_k)$ and $B^{\mathrm{T2}}(x_k)$. We now drop the subscript on x_k for ease of notation.

The resulting computation from a worker is then an evaluation of the polynomial

$$A^{\mathrm{T2}}(x)B^{\mathrm{T2}}(x) = \Theta^{\mathrm{T2}}(x)(\sum_{i=1}^{L_A}\frac{1}{a_i(x)}A_i)(\sum_{j=1}^{L_B}\frac{1}{b_j(x)}B_j) \tag{4.23}$$

$$= \sum_{i=1}^{L_A}\sum_{j=1}^{L_B}\frac{\Theta^{\mathrm{T2}}(x)}{a_i(x)b_j(x)}A_iB_j = \sum_{i=1}^{L_A}\sum_{j=1}^{L_B}\frac{\prod_{r\in[L_A]\setminus\{i\}}a_r(x)}{b_j(x)}A_iB_j \tag{4.24}$$

$$= \sum_{(i,j)\notin E}\prod_{r\notin N_B(j)\setminus\{i\}}(x-f_r)^{d_r^A}\prod_{r\in N_B(j)}(x-f_r)^{d_i^A-P_{r,j}}A_iB_j$$

$$+\sum_{(i,j)\in E}\prod_{r\notin N_B(j)}(x-f_r)^{d_r^A}\prod_{r\in N_B(j)\setminus\{i\}}(x-f_r)^{d_r^A-P_{r,j}}\frac{A_iB_j}{(x-f_i)^{P_{i,j}}}. \tag{4.25}$$

The first sum in (4.25) corresponds to the undesired computations, i.e., $(i,j)\notin E$, and the second sum in (4.25) corresponds to the desired computations, i.e $(i,j)\in E$.

We observe that the first sum in (4.25) is a polynomial in x with degree $\max_{(i,j)\notin E}|E| - d_i^A - \sum_{k=1}^{L_A}P_{k,j}$. To see this, note that $a_i(x)b_j(x)$ has a degree of $d_i^A + \sum_{k=1}^{L_A}P_{k,j}$. Since $(i,j)\notin E$, $a_i(x)$ and $b_j(x)$ do not share any roots. Thus, $\frac{\Theta(x)}{a_i(x)}$ must share the roots of $b_j(x)$ at an equal or higher multiplicity and the resulting polynomial from $\frac{\Theta(x)}{a_i(x)b_j(x)}$ must be a polynomial of degree $|E| - (d_i^A + \sum_{k=1}^{L_A}P_{k,j})$. By taking the maximum over all $(i,j)\notin E$, we conclude the degree is $\max_{(i,j)\notin E}|E| - d_i^A - \sum_{k=1}^{L_A}P_{k,j}$.

Now, let us consider the second sum in Eq. (4.25). Given a tuple $(i,j) \in E$, we write

$$\prod_{r \notin N_B(j)} (x - f_r)^{d_r^A} \prod_{r \in N_B(j) \setminus \{i\}} (x - f_r)^{d_r^A - P_{r,j}} \frac{A_i B_j}{(x - f_i)^{P_{i,j}}} \tag{4.26}$$

$$= \prod_{r \notin N_B(j)} ((x - f_i) + (f_i - f_r))^{d_r^A} \prod_{r \in N_B(j) \setminus \{i\}} ((x - f_i) + (f_i - f_r))^{d_r^A - P_{r,j}} \frac{A_i B_j}{(x - f_i)^{P_{i,j}}} \tag{4.27}$$

$$= \frac{\Delta_{i,j}(x - f_i)}{(x - f_i)^{P_{i,j}}} A_i B_j \tag{4.28}$$

$$= \Big(\frac{\zeta_{i,j,0}}{(x - f_i)^{P_{i,j}}} + \frac{\zeta_{i,j,1}}{(x - f_i)^{P_{i,j}-1}} + \cdots + \frac{\zeta_{i,j,P_{i,j}-1}}{(x - f_i)} + \zeta_{i,j,P_{i,j}} + \zeta_{i,j,P_{i,j}+1} x + \cdots + \zeta_{i,j,D_{i,j}} x^{D_{i,j} - P_{i,j}}\Big) A_i B_j \tag{4.29}$$

where Eq. (4.28) comes from the definition of $\Delta_{i,j}(x)$. We thus claim that the second expression in Eq. (4.25) results in $P_{i,j}$ rational terms and a polynomial of degree $\max_{(i,j)\in E}(D_{i,j} - P_{i,j}) = \max_{(i,j)\in E} |E| - d_i^A - \sum_{k=1}^{L_A} P_{k,j}$.

Let $T = |E| - \min_{i\in[L_A]}(d_i^A) - \min_{i\in[L_B]}(\sum_{k=1}^{L_A} P_{k,j})$. Applying the expression from Eq. (4.29) and combining the two expressions in (4.25), we arrive at the following equation

$$A^{\mathrm{T2}}(x) B^{\mathrm{T2}}(x) = \sum_{(i,j)\in E} \frac{1}{(x - f_i)^{P_{i,j}}} \sum_{r \in N_A(i): P_{i,r} \geq P_{i,j}} \zeta_{i,r,P_{i,r}-P_{i,j}} A_i B_r + \sum_{r=0}^{T} I_r x^r \tag{4.30}$$

where $\{I_r : r \in \{0, 1, \ldots, T\}\}$ are arbitrary interference terms.

By Lemma 1, $A^{\mathrm{T2}}(x) B^{\mathrm{T2}}(x)$ can be interpolated from $2|E| - \min_{i\in[L_A]}(d_i^A) - \min_{i\in[L_B]}(\sum_{k=1}^{L_A} P_{k,j}) + 1$ evaluations and we can acquire the coefficients associated with the rational and polynomial terms. Now, we show that $\{A_i B_j : (i,j) \in S\}$ can be extracted from the coefficients of $A^{\mathrm{T2}}(x) B^{\mathrm{T2}}(x)$.

We observe that in Eq. (4.30) the coefficients of the rational terms with a pole at f_i contain only the computations where A_i is involved in. As such, we fix the index i and focus on the subspace generated by the powers of $\frac{1}{(x-f_i)}$. For convenience, let $d = d_i^A$. We define the ordered index set $j_1, j_2, \ldots, j_d$ for the non-zero values of $P_{i,j}$ where $j_a < j_b \iff P_{i,a} < P_{i,b}$. Note that $P_{i,j_1} = 1$ and $P_{i,j_d} = d$. We can write the rational terms associated with the root

f_i as follows

$$\sum_{s=1}^{d} \frac{1}{(x-f_i)^{P_{i,j_s}}} \sum_{r=s}^{d} \zeta_{i,j,r-s} A_i B_{j_r}. \tag{4.31}$$

Let $Y_s = \sum_{r=s}^{d} \zeta_{i,j,r-s} A_i B_r$. In matrix notation, we write Y_s as

$$\begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_d \end{bmatrix} = \underbrace{\begin{bmatrix} \zeta_{i,j_1,0} & \zeta_{i,j_2,1} & \zeta_{i,j_3,2} & \cdots & \zeta_{i,j_d,d-1} \\ & \zeta_{i,j_2,0} & \zeta_{i,j_3,1} & \cdots & \zeta_{i,j_1,d} \\ & & \ddots & \vdots & \vdots \\ & & & \zeta_{i,j_{d-1},0} & \zeta_{i,j_{d-1},1} \\ & & & & \zeta_{i,j_d,0} \end{bmatrix}}_{V_i} \begin{bmatrix} A_i B_{j_1} \\ A_i B_{j_2} \\ \vdots \\ A_i B_{j_d} \end{bmatrix}. \tag{4.32}$$

To extract the desired computations, we need to show that V_i is invertible. Since V_i is an upper-triangular matrix, it is sufficient to show that all elements along the diagonal are non-zero. This is clearly true since

$$\zeta_{i,j,0} = \Delta_{i,j}(0) = \prod_{r \notin N_B(j)} (f_i - f_r)^{d_r^A} \prod_{r \in N_B(j) \setminus \{i\}} (f_i - f_r)^{d_r^A - P_{r,j}} \neq 0. \tag{4.33}$$

Thus, we have constructed FCSA-T2 codes that achieve the desired recovery threshold which completes the proof. □

Now that we have shown how to construct FCSA-T2 codes, we provide the left power assignment matrices for the examples in the previous section which are

$$P_1^{\mathrm{T2}} = \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix}, P_2^{\mathrm{T2}} = \begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix}.$$

Remark. *Since FCSA-T2 codes are also shape-preserving codes, we can clearly apply Lemma 2 to FCSA-T2 codes and claim that they are within a factor of two optimal for all power*

$$\arg\max_{p_{i,j}} \quad \min_{j\in[L_B]} \left(\sum_{i=1}^{L_A} p_{i,j}\right) \tag{4.34a}$$

$$\text{s.t.} \quad p_{i,j} = 0, \qquad (i,j) \notin E, \tag{4.34b}$$

$$p_{i,j} \in \{1, 2, \ldots, d_i^A\}, \qquad (i,j) \in E, \tag{4.34c}$$

$$p_{i,j} \neq p_{i,k}, \qquad j \neq k\ , (i,j)\ , (i,k) \in E, \tag{4.34d}$$

$$p_{i,j} : \text{Power associated to edge } (i,j)$$

Figure 4.1: Optimization problem for power assignment matrix of FCSA-T2 codes.

assignment matrices P. This indicates that there may exist codes that have a better optimality gap and FCSA-T2 codes is a good first attempt at achieving them.

Additionally, FCSA-T2 codes have a lighter constraint on the size of the field, requiring only that $|\mathbb{F}| > L_A + K$ while FCSA-T1 codes require $|\mathbb{F}| > |E| + K$. Since $L_A < |E|$, FCSA-T2 codes enjoy the benefits of using a smaller field size such as the reduction in complexity of arithmetic operations.

4.3 Optimization of FCSA-T2 codes

Now that we have shown how to construct FCSA-T2 codes from a power assignment matrix P, we need to optimize P to minimize the recovery threshold as much as possible. To accomplish this, we formulate the problem as a mixed-integer linear program for which there are many known solvers for [41, 42].

As shown in Eq. (4.15), the recovery threshold is $2|E| - \min_{i\in[L_A]}(d_i^A) - \min_{j\in[L_B]}(\sum_{i=1}^{L_A} P_{i,j})$. To improve the recovery threshold, we need to find the optimal P that maximizes $\min_{j\in[L_B]}(\sum_{i=1}^{L_A} P_{i,j})$ while still being a power assignment matrix. Using the definition of P, we define our optimization problem in Fig. 4.1. In its current form, the optimization problem is impractical to

solve which requires a re-formulation.

We can reformulate the program in Fig. 4.1 into a zero-one linear program where the only integer variables are binary. To do this, we note that $p_{i,j} \in \{1, 2, \ldots, d_i^A\}$ can be replaced by $p_{i,j} = \sum_{k=1}^{d_i^A} k \cdot y_{i,j,k}$ where $y_{i,j,k}$ are binary 0-1 variables indicating that $p_{i,j}$ has the value of k. To ensure that $p_{i,j}$ takes on one of these values, we set $1 = \sum_{k=1}^{d_i^A} y_{i,j,k}$ for $(i,j) \in E$. Additionally, we handle the not-equals constraint in Fig. 4.1 by adding the constraint $1 = \sum_{j=1}^{d_i^A} y_{i,j,k},\ i \in [L_B],\ k \in \{1, 2, \ldots, d_i^A\}$. Thus, the final reformulated mixed-integer linear program is shown in Fig. 4.2.

It is well known that mixed-integer programming is generally NP-hard. Therefore, to our knowledge, the best approach to get an optimal solution for Fig. 4.2 is applying a branch and cut methodology [43]. In subsequent sections, we use the mixed integer optimization solver *coin-or cbc* that uses branch and cut methods [42]. While applying this general approach is sufficient for small bipartite graphs, applications of FCSA-T2 codes to large graphs requires specialized optimization techniques which is a possible focus for future study.

4.4 Complexity Analysis of FCSA-T2 codes

Computational and Communication Complexity: Since FCSA-T2 codes are shape-preserving codes, we have the upload costs as $U_A = \alpha\beta$ and $U_B = \beta\gamma$ and the download cost as $D_C = \alpha\gamma$. Additionally, the computational complexity at each worker is $\mathcal{O}(\alpha\beta\gamma)$.

Encoding and Decoding Complexity: The encoding complexity depends on whether the optimization of P is performed offline or online. If performed offline, the encoding complexity is the same as for FCSA-T1 codes which is $\mathcal{O}(KL_A\alpha\beta)$ for Source $\mathcal{A}$ and $\mathcal{O}(KL_B\beta\gamma)$ for Source $\mathcal{B}$. If performed online, the encoding complexity is dominated by the time needed to optimize for P which requires further study into the mixed-integer linear program that we presented. There are a plethora of techniques left that can be used to speed up the optimization [44].

$$
\begin{aligned}
\underset{p_{i,j},\, y_{i,j,k},\, z}{\arg\max} \quad & z && \text{(4.35a)}\\
\text{s.t.} \quad & z \le \sum_{i=1}^{L_A} p_{i,j}, \quad j \in [L_B], && \text{(4.35b)}\\
& p_{i,j} = 0, \quad (i,j) \notin E, && \text{(4.35c)}\\
& p_{i,j} = \sum_{k=1}^{d_i^A} k \cdot y_{i,j,k}, \quad (i,j) \in E, && \text{(4.35d)}\\
& 1 = \sum_{k=1}^{d_i^A} y_{i,j,k}, \quad (i,j) \in E, && \text{(4.35e)}\\
& 1 = \sum_{j=1}^{d_i^A} y_{i,j,k}, \quad i \in [L_B],\ k \in \{1,2,\ldots,d_i^A\}, && \text{(4.35f)}\\
& z \in \mathbb{R}, p_{i,j} \in \mathbb{R}, && \text{(4.35g)}\\
& y_{i,j,k} \in \{0,1\}, \quad (i,j) \in E,\ k \in \{1,2,\ldots,d_i^A\}, && \text{(4.35h)}\\
& p_{i,j} : \text{Power associated to edge } (i,j) && ,\\
& y_{i,j,k} : \text{Binary indicator that edge } (i,j) \text{ has power } k &&
\end{aligned}
$$

Figure 4.2: Linear Binary Optimization program for FCSA-T2 codes. This is a reformulation of the program in Fig. 4.2.

The decoding complexity involves two steps. First is inverting a $R_{\text{FCSA-T2}} \times R_{\text{FCSA-T2}}$ confluent Cauchy-Vandermonde matrix which is the same step as for FCSA-T1 codes. As for FCSA-T1 codes, this step has a complexity of $\widetilde{\mathcal{O}}(\alpha\gamma \cdot R_{\text{FCSA-T2}} \log^2 R_{\text{FCSA-T2}})$. The second step involves inverting L_A upper triangular matrices of sizes $\{d_i^A : i \in [L_A]\}$. A standard forward substitution algorithm can invert an upper-triangular matrix of size d using $\mathcal{O}(d^2)$ operations. As such, inverting all the upper-triangular matrices has a complexity of $\mathcal{O}(\sum_{i\in[L_a]}(d_i^A)^2) \le \mathcal{O}((L_A)^3)$. Hence, the overall decoding complexity is upper-bounded by $\widetilde{\mathcal{O}}(\alpha\gamma \cdot R_{\text{FCSA-T2}} \log^2 R_{\text{FCSA-T2}} + (L_A)^3)$.

4.5 FCSA-T2 performance on $\mathcal{G}_{L_A,L_B,p}$

Similar to FCSA-T1 codes, we demonstrate the average improvement allowed by FCSA-T2 codes for random bipartite graphs. Recall that the average improvement is defined as $\mathbb{E}[R_{Naive} - R_{\text{FCSA-T1}}]$ where R_{Naive} is the recovery threshold of either LCC or CSA codes. While we cannot give a definitive growth rate, we note that FCSA-T2 must perform at least as well as FCSA-T1 codes. As such, the average improvement must at least grow linearly in terms of the number of vertices as shown in Lemma 3. To provide the best improvement for FCSA-T2 codes, we run the optimization program over left and right power assignment matrices and choose the best recovery threshold among the two.

Similar to the simulations of FCSA-T1 codes, we provide simulations for the average improvement for various scenarios. The first scenario is simulating the average improvement across varying L_A for different values of p with $L_B = 5$. We plot the improvement for both Type 1 and Type 2 codes. The results of this experiment can be seen in Fig. 4.3. The first thing we notice is that clearly FCSA-T2 codes outperform FCSA-T1 codes for all scenarios, in terms of the recovery threshold. Interestingly, the average improvement grows linearly in terms of the number of vertices for both Type 1 and Type 2 codes. We also note how the gap between Type 1 and Type 2 grows wider for increasing values of L_A and p. This indicates that FCSA-T2 codes better utilize the redundancy in computation to further reduce the recovery threshold.

Our second simulation scenario demonstrates the average improvement for different graph ensembles with the same expected number of edges. By keeping the average number of edges the same, all the ensembles have the same lower bound for the optimal recovery threshold as guaranteed by Lemma 2. As such, we simulate the performance over graph ensembles $\mathcal{G}_{L_A,L_B,p}$ with fixed pairs of (L_A, L_B) where $L_A L_B$ is kept constant. The results of this simulation can be found in 4.4. Again, we plot the improvement for both Type 1 and Type 2 codes. Similar to the previous scenario, FCSA-T2 codes have the best recovery threshold with the gap widening

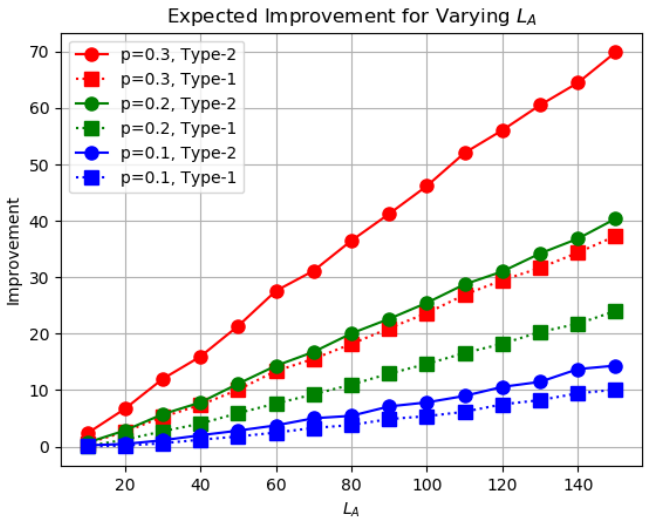


Figure 4.3: Shows the average improvement for the recovery threshold using both FCSA-T1 and FCSA-T2 codes across varying L_A, fixed values for p, and $L_B = 5$.

as the graph becomes denser. We note that the $(100, 5)$ pair has the best improvement while the $(25, 20)$ pair has the worst improvement. We speculate a similar rationale as for FCSA-T1 codes that the pair $(25, 20)$ allows for less concentration of computations on a single vertex which reduces the redundancy it can be used for. These simulations suggest that there may be a graph construct that provides better bounds for the optimal recovery threshold.

Finally, we present the average recovery thresholds for FCSA-T1 codes, FCSA-T2 codes, and the codes described in chapter 2 to demonstrate how FCSA codes better approach the optimal lower bound as shown in Lemma 2. We take the lower bound as the average number of edges in the graph, i.e., $L_A L_B p$. In Fig. 4.5, we demonstrate the average recovery thresholds for a similar scenario as in Fig. 4.3 where L_A is varying while L_B and p are fixed. Observe that as the number of vertices increases, FCSA codes perform strictly better than Polynomial, LCC, and CSA codes. Additionally, we see that FCSA-T2 codes get much closer to the optimal lower bound. In Fig. 4.6, we demonstrate the recovery threshold using a similar scenario to Fig. 4.4 where p is varying while L_A and L_B are fixed. We see here that

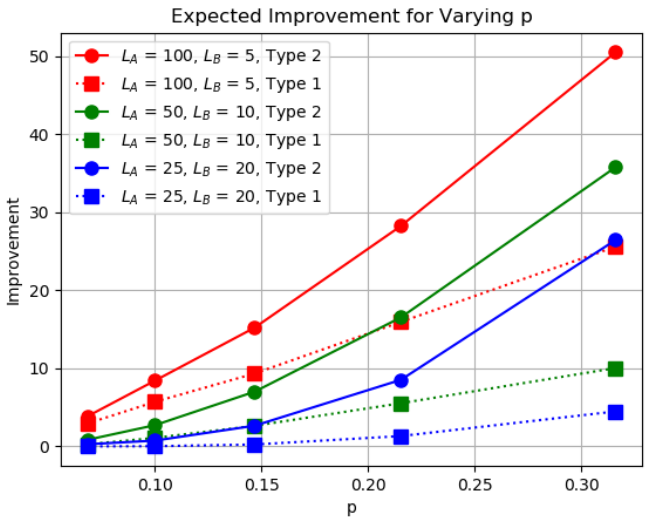


Figure 4.4: Shows the average improvement for the recovery threshold using both FCSA-T1 and FCSA-T2 codes across varying p and fixed pairs of (L_A, L_B).

Polynomial codes have a fixed recovery threshold regardless of the value of p which is simply $L_A L_B$. As p increases, Polynomial codes will eventually have the lowest recovery threshold and will attain the optimal lower bound when $p = 1$. Before that cross-over point, FCSA codes have a much smaller recovery threshold than all the other codes. Additionally, we see that the gap between FCSA codes and LCC or CSA codes gets much wider as p increases, as shown in the previous experiments. While both of these graphs demonstrate how FCSA codes strictly improve on the recovery threshold in comparison to the other codes, we still see that the gap between FCSA codes and the optimal lower bound diverges as the graph gets larger or denser. As discussed in the previous experiments, further research is required to determine whether this gap is insurmountable or not.

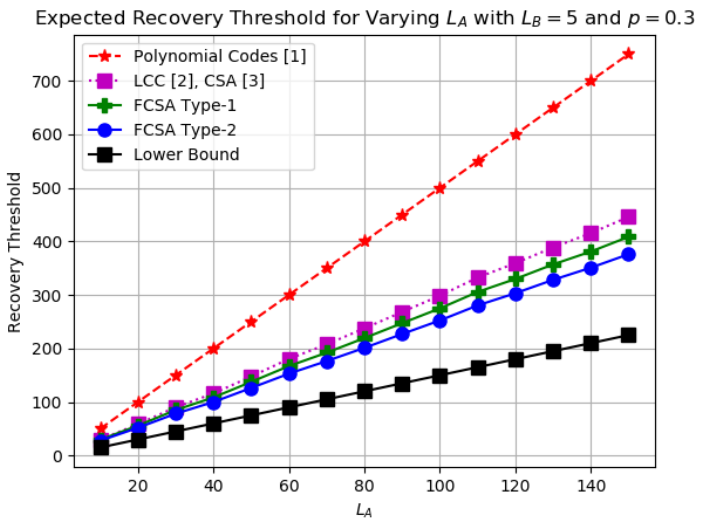


Figure 4.5: Recovery Threshold for different VCDBMM codes given varying values of L_A with $L_B = 5$ and $p = 0.3$.

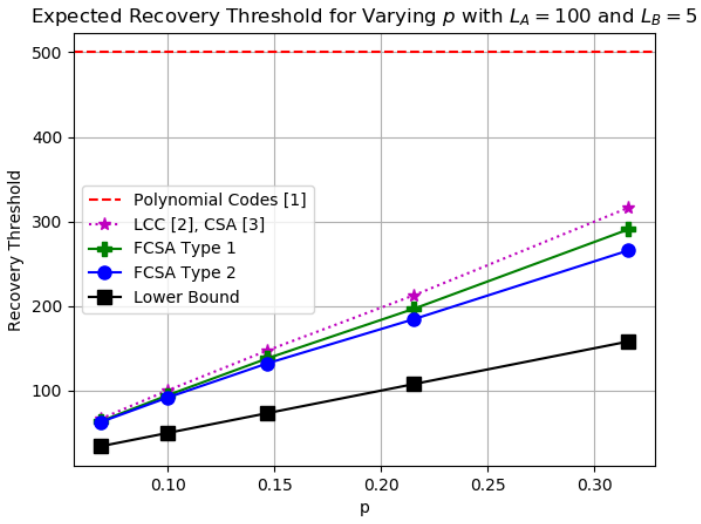


Figure 4.6: Recovery Threshold for different VCDBMM codes given varying values of p with $L_A = 100$ and $L_B = 5$.

CHAPTER 5

Upscaling by Grouping

Consider a scenario where the matrices in $\mathcal{A}$ and $\mathcal{B}$ are sufficiently small such that the computation complexity at each worker is small. In this case, the major bottle-neck is the number of responses that the fusion node needs to receive since some workers may be stragglers. In [3], researchers grouped the matrix computations such that each group was separately encoded using Cauchy encoding and summed up the resulting computations in each worker node. This results in increased upload cost and computation complexity while decreasing the recovery threshold without affecting the download cost. We can apply a similar grouping as [3] within the framework of the VCDBMM problem. We demonstrate how grouping can be applied to both variants of FCSA codes.

The grouping in [3] can be represented by an edge partition of G. Recall that an edge partition $G^{\mathcal{P}} = \{G_1 = (U_1 \cup V_1, E_1), G_2 = (U_2 \cup V_2, E_2), \ldots, G_Q = (U_Q \cup V_Q, E_Q)\}$ is a collection of Q sub-graphs with $E_i \cap E_j = \emptyset,\ \forall i, j \in [Q]$ and $\cup_{q\in[Q]} E_q = E$. We can consider each sub-graph as a grouping of computations. Additionally, recall that $N_A^q(i)$ is the set of adjacent vertices to the left vertex i within the sub-graph G_q with $d_{i,q}^A = |N_A^q(i)|$. Similarly, $N_B^q(j)$ is the set of adjacent vertices to the right vertex j within the sub-graph G_q with $d_{j,q}^B = |N_B^q(j)|$.

5.1 Grouping with FCSA codes

We first demonstrate grouping using FCSA-T1 codes which are easier to apply grouping to. The recovery threshold and computational complexity trade-off is characterized by the following theorem.

Theorem 3. *Let $G = (U \cup V, E)$ be a VCDBMM bipartite graph. Let $G^{\mathcal{P}} = \{G_1 = (U_1 \cup V_1, E_1), G_2 = (U_2 \cup V_2, E_2), \ldots, G_Q = (U_Q \cup V_Q, E_Q)\}$ be an edge partition of G where the number of partitions is Q. Assume that $|\mathbb{F}| > |E| + K$. Given G^P, Grouped FCSA-T1 (GFCSA-T1) codes can achieve a recovery threshold of*

$$R_{GFCSA\text{-}T1} = |E| + \max_{q \in [Q]} (|E_q| - \min_{i \in U_q} d^A_{i,q} - \min_{j \in V_q} d^B_{j,q}) + 1 \tag{5.1}$$

with worker complexity $\mathcal{O}(Q \cdot \alpha\beta\gamma)$ while keeping the exact same download costs D_C as with no grouping.

Proof. To prove Theorem 3, we now construct Grouped FCSA-T1 codes.

For each edge $e \in E$, we associate a distinct element f_e from $\mathbb{F}$. We select K additional distinct elements from $\mathbb{F}$ and represent them as $x_1, x_2, \ldots, x_K$. We assume that all $G \in G^{\mathcal{P}}$ are pruned of all isolated vertices.

First, we create individual FCSA-T1 codes for each sub-graph G_q using the elements $\{f_e : e \in E\}$. Hence, we create encoding polynomials $A^{\mathrm{T1}}_q(x)$ and $B^{\mathrm{T1}}_q(x)$ for each G_q using the construction of FCSA-T1 codes as defined in chapter 3.

For the k^{th} worker, we construct $\widetilde{A}_k$ and $\widetilde{B}_k$ as

$$\widetilde{A}_k = \begin{bmatrix} A^{\mathrm{T1}}_1(x_k) & A^{\mathrm{T1}}_2(x_k) & \cdots & A^{\mathrm{T1}}_Q(x_k) \end{bmatrix}, \widetilde{B}_k = \begin{bmatrix} B^{\mathrm{T1}}_1(x_k) \\ B^{\mathrm{T1}}_2(x_k) \\ \vdots \\ B^{\mathrm{T1}}_Q(x_k) \end{bmatrix} \tag{5.2}$$

where $\widetilde{A}_k \in \mathbb{F}^{\alpha \times Q\beta}$ and $\widetilde{B}_k \in \mathbb{F}^{Q\beta \times Q\gamma}$. Thus, the output $\widetilde{C}_k$ at the k^{th} worker is

$$\widetilde{C}_k = \widetilde{A}_k \widetilde{B}_k = \sum_{q=1}^{Q} A_q^{\text{T1}}(x_k) B_q^{\text{T1}}(x_k) \triangleq C(x_k). \tag{5.3}$$

Note that $\widetilde{C}_k \in \mathbb{F}^{\alpha \times \gamma}$ so the download cost is $D_C = \alpha\gamma$. Additionally, the complexity of multiplying $\widetilde{A}_k$ and $\widetilde{B}_k$ is $\mathcal{O}(Q \cdot \alpha\beta\gamma)$. We now drop the subscript k in x_k.

Recall from Eq. 3.11 that

$$A_q^{\text{T1}}(x) B_q^{\text{T1}}(x) = \sum_{(i,j)\in E_q} \frac{\zeta_{i,j}}{(x - f_{(i,j)})} A_i B_j + \sum_{r=0}^{T_q} I_r x^r \tag{5.4}$$

where $\{I_r : r \in \{0, 1, \ldots, T\}\}$ are arbitrary interference terms, $\{\zeta_{i,j} : (i,j) \in E\}$ are non-zero constants, and $T_q = |E_q| - \min_{i \in U_q} d_i^A - \min_{j \in V_q} d_j^B$. By definition, $E_i \cap E_j = \emptyset$ and $\cup_{i \in [Q]} E_i = E$. Therefore, we write $C(x)$ as

$$C(x) = \sum_{(i,j)\in E} \frac{\zeta_{i,j}}{(x - f_{(i,j)})} A_i B_j + \sum_{r=0}^{\max_{q\in[Q]} T_r} I_r x^r \tag{5.5}$$

where again $\{I_r : r \in \{0, 1, \ldots, T\}\}$ are arbitrary interference terms. Note that $\max_{q\in[Q]} T_r = \max_{q\in[Q]}(|E_q| - \min_{i\in U_q} d_i^A - \min_{j\in V_q} d_j^B)$. We observe that $\max_{q\in[Q]} T_r$ may take the value of -1 which indicates that there are no polynomial terms in $C(x)$.

By Lemma 1, we interpolate $C(x)$ from $|E| + \max_{q\in[Q]}(|E_q| - \min_{i\in U_q} d_i^A - \min_{j\in V_q} d_j^B) + 1$ evaluations and acquire $\{\zeta_{i,j} A_i B_j : (i,j) \in S\}$. The desired computations are then extracted and re-scaled appropriately. Thus, Grouped FCSA-T1 codes achieve the recovery threshold of $R_{\text{GFCSA-T1}}$ with the desired download cost and worker complexity. □

Now, we demonstrate how to apply grouping to FCSA-T2 codes. The construction follows similar logical lines as for Grouped FCSA-T1 codes.

Theorem 4. *Let $G = (U \cup V, E)$ be a VCDBMM bipartite graph and let $G^{\mathcal{P}} = \{G_1 = (U_1 \cup V_1, E_1), G_2 = (U_2 \cup V_2, E_2), \ldots, G_Q = (U_Q \cup V_Q, E_Q)\}$ be an edge partition of G where the number of partitions is Q. Let $P^1, P^2, \ldots, P^Q$ be power assignment matrices for each*

sub-graph. Assume that $|\mathbb{F}| > \sum_{q\in[Q]} |U_q| + K$*. Given* G^P*, grouped FCSA-T2 (GFCSA-T2) codes can achieve a recovery threshold of*

$$R_{GFCSA\text{-}T2} = |E| + \max_{q\in[Q]}(|E_q| - \min_{i\in U_q}(d^A_{i,q}) - \min_{j\in V_q} \sum_{i\in N^q_B(j)} P^q_{i,j}) + 1 \tag{5.6}$$

with worker complexity $\mathcal{O}(Q \cdot \alpha\beta\gamma)$ *while keeping the exact same download costs* D_C *as with no grouping.*

Proof. To prove Theorem 4, we now construct Grouped FCSA-T2 codes.

Given a sub-graph G_q, each vertex $i \in [U_q]$ is assigned a distinct element $f_{i,q}$ from $\mathbb{F}$. We select K additional distinct elements from $\mathbb{F}$ and represent them as $x_1, x_2, \ldots, x_K$. This is where the constraint $|\mathbb{F}| > \sum_{q\in[Q]} |U_q| + K$ comes from.

Now, we create individual FCSA-T2 codes for each sub-graph G_q. The FCSA-T2 code for G_q uses the elements $\{f_{i,q} : i \in U_q\}$ and power assignment matrix P^q. Hence, we create encoding polynomials $A^{\mathrm{T2}}_q(x)$ and $B^{\mathrm{T2}}_q(x)$ for each G_q using the construction of FCSA-T2 codes as defined in chapter 4.

For the k^{th} worker, we construct $\widetilde{A}_k$ and $\widetilde{B}_k$ as

$$\widetilde{A}_k = \begin{bmatrix} A^{\mathrm{T2}}_1(x_k) & A^{\mathrm{T2}}_2(x_k) & \cdots & A^{\mathrm{T2}}_Q(x_k) \end{bmatrix}, \widetilde{B}_k = \begin{bmatrix} B^{\mathrm{T2}}_1(x_k) \\ B^{\mathrm{T2}}_2(x_k) \\ \vdots \\ B^{\mathrm{T2}}_Q(x_k) \end{bmatrix} \tag{5.7}$$

where $\widetilde{A}_k \in \mathbb{F}^{\alpha\times Q\beta}$ and $\widetilde{B}_k \in \mathbb{F}^{Q\beta\times Q\gamma}$. Thus, the output $\widetilde{C}_k$ at the k^{th} worker is

$$\widetilde{C}_k = \widetilde{A}_k\widetilde{B}_k = \sum_{q=1}^{Q} A^{\mathrm{T2}}_q(x_k)B^{\mathrm{T2}}_q(x_k) \triangleq C(x_k). \tag{5.8}$$

Note that $\widetilde{C}_k \in \mathbb{F}^{\alpha\times\gamma}$ so the download cost is $D_C = \alpha\gamma$. Additionally, the complexity of multiplying $\widetilde{A}_k$ and $\widetilde{B}_k$ is $\mathcal{O}(Q \cdot \alpha\beta\gamma)$. We can now drop the subscript k in x_k.

Recall from Eq. 4.30 that

$$A_q^{\mathrm{T2}}(x)B_q^{\mathrm{T2}}(x) = \sum_{(i,j)\in E_q} \frac{1}{(x-f_{i,q})^{P_{i,j}^q}}(\zeta_{i,j}^q A_i B_j + \sum_{r\in N_A^q(i):P_{i,r}>P_{i,j}} \eta_{i,j,r}^q A_i B_r) + \sum_{r=0}^{T_q} I_r x^r \quad (5.9)$$

where $\{I_r : r \in \{0,1,\ldots,T\}\}$ are arbitrary interference terms, $\{\zeta_{i,j}^q : q \in [Q], (i,j) \in E\}$ are non-zero constants, $\{\eta_{i,j,r} : q \in [Q], (i,j) \in [Q], r \in N_A^q(i)\}$ are arbitrary constants, and $T_q = |E_q| - \min_{i\in U_q} d_{i,q}^A - \min_{j\in V_q} \sum_{i\in N_B^q(j)} P_{i,j}^q$.

Therefore, we write $C(x)$ as

$$C(x) = \sum_{q\in[Q]} (\sum_{(i,j)\in E_q} \frac{1}{(x-f_{i,q})^{P_{i,j}^q}}(\zeta_{i,j}^q A_i B_j + \sum_{r\in N_A^q(i):P_{i,r}>P_{i,j}} \eta_{i,j,r}^q A_i B_r)) + \sum_{r=0}^{\max_{q\in[Q]} T_q} I_r x^r \quad (5.10)$$

where again $\{I_r : r \in \{0,1,\ldots,T\}\}$ are arbitrary interference terms.

Note that $\max_{q\in[Q]} T_r = \max_{q\in[Q]} \left(|E_q| - \min_{i\in U_q} d_{i,q}^A - \min_{j\in V_q} \sum_{i\in N_B^q(j)} P_{i,j}^q\right)$. By Lemma 1, we interpolate $C(x)$ from $|E| + \max_{q\in[Q]} \left(|E_q| - \min_{i\in U_q} d_i^A - \min_{j\in V_q} \sum_{i\in N_B^q(j)} P_{i,j}^q\right) + 1$ evaluations. Now, we show that $\{A_i B_j : (i,j) \in S\}$ can be extracted from the coefficients of $C(x)$.

We observe that in Eq. (5.10) the coefficients of the rational terms with a pole at $f_{i,q}$ contain only the computations $A_i B_j$ for $j \in V_q$. As such, we fix i and q and focus on the subspace generated by the powers of $\frac{1}{(x-f_{i,q})}$. For convenience, we let $d = d_{i,q}^A$. We define the ordered index set $j_1, j_2, \ldots, j_d$ for the non-zero values $\{P_{i,j}^q : j \in N_A^q(i)\}$ where $j_a < j_b \iff P_{i,a}^q \leq P_{i,b}^q$. Note that $P_{i,j_1}^q = 1$ and $P_{i,j_d}^q = d$. We can write the rational terms associated with the root $f_{i,q}$ as follows

$$\sum_{s=1}^{d} \frac{1}{(x-f_{i,q})^{P_{i,j_s}^q}}(\zeta_{i,j}^q A_i B_{j_s} + \sum_{r=s+1}^{d} \eta_{i,j,r}^q A_i B_{j_r}). \quad (5.11)$$

Let $Y_s = \zeta_{i,j}^q A_i B_{j_s} + \sum_{r=s+1}^{d} \eta_{i,j,r}^q A_i B_{j_r}$. Using matrix notation, $\{Y_s : s \in [d]\}$ can be

written as

$$\begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_d \end{bmatrix} = \underbrace{\begin{bmatrix} \zeta_{i,j_1,0} & * & * & \dots & * \\ & \zeta_{i,j_2,0} & * & \dots & * \\ & & \ddots & \vdots & \vdots \\ & & & \zeta_{i,j_{d-1},0} & * \\ & & & & \zeta_{i,j_d,0} \end{bmatrix}}_{V_i} \begin{bmatrix} A_i B_{j_1} \\ A_i B_{j_2} \\ \vdots \\ A_i B_{j_d} \end{bmatrix}. \tag{5.12}$$

where $*$ are arbitrary terms. By construction of FCSA-T2 codes, the diagonal elements of V_i are non-zero which guarantees that V_i is invertible. Thus, we acquire $\{A_i B_j : j \in V_q\}$ for $q \in [Q]$ and $i \in [U_q]$. Since $\cup_{i\in[Q]} E_i = E$, we acquire all the desired computations $\{A_i B_j : (i,j) \in \mathcal{S}\}$. Hence, Grouped FCSA-T2 codes achieve the recovery threshold of $R_{\text{GFCSA-T2}}$ with the desired download cost and computational complexity. □

Remark. *We note that the previous construction of GFCSA-T2 codes implicitly stipulated that all power assignment matrices P^q must be left assignment matrices. From the code constructions, it clear that we did not rely on the fact that each P^q is only a left power assignment matrix and, in fact, can choose the matrix to be either left or right focused depending on which one gives a better recovery threshold. For brevity, we omit the construction where P^q may be either a left or right power assignment matrix but note that all further simulations will which choose the best of the two for each sub-graph.*

In general, we cannot make any claims about $R_{\text{GFCSA-T1}}$ and $R_{\text{GFCSA-T2}}$ based purely on the value of Q except in two extreme cases. The obvious case is $Q = 1$ which degenerates the codes to standard FCSA-T1 and FCSA-T2 codes whose recovery threshold is independent of Q. Now, we focus on the more interesting extreme. Without loss of generality, assume that $L_A < L_B$ and let $Q = L_A$. We define an edge partition $G^{\mathcal{P}}$ by assigning each left vertex and its adjacent edges into its own unique sub-graph. Thus, we let G_q be the sub-graph that contains the left vertex q and all of its adjacent vertices. Since each sub-graph contains

exactly one left vertex, $d^A_{q,q} = |E_q|$ and $d^A_{i,q} = 0$ for $i \in [L_A] \setminus \{q\}$. Additionally, $d^B_{j,q} = 1$ if $(j,q) \in E_q$ else $d^B_{j,q} = 0$. For Grouped FCSA-T1 codes, we write $R_{\text{GFCSA-T1}}$ as

$$R_{\text{GFCSA-T1}} = |E| + \max_{q\in[Q]}(|E_q| - \min_{i\in U_q} d^A_{i,q} - \min_{j\in V_q} d^B_{j,q}) + 1 \tag{5.13}$$

$$= |E| + \max_{q\in[Q]}(|E_q| - |E_q| - 1) + 1 = |E|. \tag{5.14}$$

Note that the proof for Lemma 2 only relied on the download cost being fixed. Since GFCSA-T1 codes have the same download costs as shape-preserving codes, we achieve the optimal recovery threshold by setting $Q = L_A$ and using this quick partitioning method. Hence, $Q = \min(L_A, L_B)$ is another extreme that upper bounds the values that Q should take. Similarly, this edge partition also achieves the minimum recovery threshold $|E|$ for $R_{GFCSA-T2}$ by noting that $\min_{j\in V_q} \sum_{i\in N^q_B(j)} P^q_{i,j} = \min_{j\in V_q} P^q_{i,j} = 1$.

5.2 Complexity Analysis of Grouped FCSA codes

Computational and Communication Complexity: Let Q be the number of partitions. By construction, GFCSA codes have a computational complexity of $\mathcal{O}(Q \cdot \alpha\beta\gamma)$ at each worker and an upload cost of $D_C = \alpha\gamma$. Since each source node sends Q matrices of the same shape as the input matrices, the upload costs are $U_A = Q\alpha\beta$ and $U_B = Q\beta\gamma$. Observe that all the costs except D_C are scaled by a factor of Q in comparison to FCSA codes.

Encoding and Decoding Complexity: Let $G^{\mathcal{P}} = \{G_1 = (U_1 \cup V_1, E_1), G_2 = (U_2 \cup V_2, E_2), \ldots, G_Q = (U_Q \cup V_Q, E_Q)\}$ be the edge partition. For GFCSA-T1 codes and GFCSA-T2 codes that are optimized offline, the encoding complexity is dominated by scaling and summing up the matrices. Each partition is encoded using the vertices in that sub-graph. For the partition G_q and a single worker, Source $\mathbb{A}$ has to scale $|U_q|$ matrices of size $\alpha \times \beta$ and then sum them up which has a complexity of $\mathcal{O}(|U_q|\alpha\beta)$. Summing for all partitions and all workers, the total encoding complexity for Source $\mathbb{A}$ in constructing the coded matrices is $\mathcal{O}(K \cdot \sum_{q\in[Q]} |U_q|\alpha\beta)$. By a similar argument, the encoding complexity for Source $\mathbb{B}$ is

$\mathcal{O}(K \cdot \sum_{q\in[Q]} |V_q|\beta\gamma)$. For the online complexity of optimizing GFCSA-T2 codes, we refer the reader to section 4.4 where we discuss optimizing FCSA-T2 codes. We do note that each partition can be optimized individually which can be parallelized for further improvement. Additionally, if the sizes of the partitions are significantly smaller than the original graph, then we would expect an exponential improvement in the optimization speed.

The decoding complexity of GFCSA-T1 codes is identical to FCSA-T1 codes because they only involve inverting a Confluent Cauchy-Vandermonde matrix. Similarly as in section 3.2, the decoding complexity of GFCSA-T1 codes is $\widetilde{\mathcal{O}}(\alpha\gamma \cdot R_{\text{GFCSA-T1}} \log^2 R_{\text{FCSA-T1}})$. Likewise for GFCSA-T2 codes, the decoding also involves inverting a Confluent Cauchy-Vandermonde matrix and a set of upper-triangular matrices. The only caveat is that GFCSA-T2 codes have to invert upper-triangular matrices with sizes $\{d^A_{i,q} : q \in [Q], i \in [U_q]\}$. As such, the overall decoding complexity for GFCSA-T2 codes is $\widetilde{\mathcal{O}}(\alpha\gamma \cdot R_{\text{GFCSA-T1}} + \sum_{q\in[Q]} \sum_{i\in[U_q]} (d^A_{i,q})^2)$.

5.3 Random Partitioning by Vertices

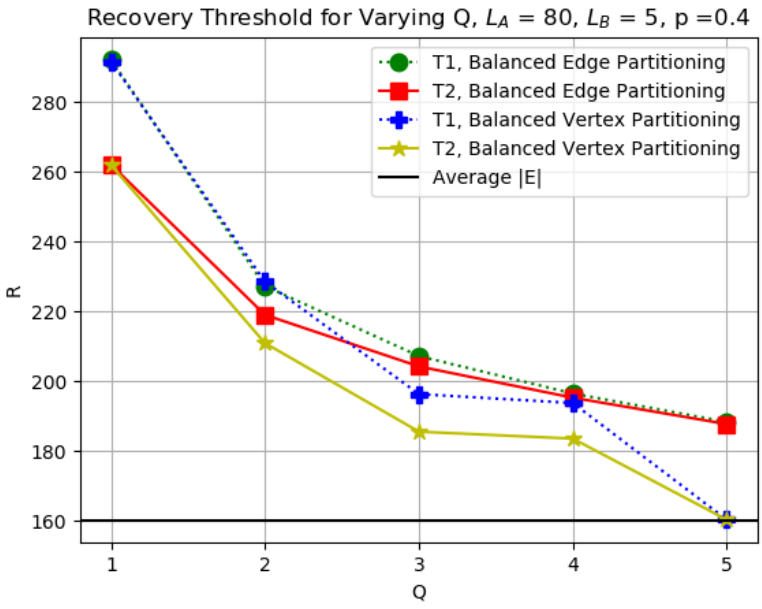
Theorems 3 and 4 demonstrate how to construct Grouped FCSA codes given an edge partition of a bipartite graph G. The only concern left is how to determine a good partition. Theoretically it is possible to devise an optimization problem to determine an optimal partition but we do not expect it to be polynomial time solvable due to the non-linear nature of the task. Instead, we demonstrate the efficacy of using a random balanced edge partition. The major benefit of using a random partition is the low complexity to create the partition which only negligibly affects the online encoding complexity. We now describe two methods of partitioning and provide experimental results for both.

Assume that we want to create Q partitions of a graph $G = (U \cup V, E)$, i.e., $G^{\mathcal{P}} = \{G_1 = (U_1 \cup V_1, E_1), G_2 = (U_2 \cup V_2, E_2), \ldots, G_Q = (U_Q \cup V_Q, E_Q)\}$ such that $\cup_{q\in[Q]} G_q = G$. Without loss of generality, assume that Q divides $|E|$. One simple strategy is to evenly split the edges of the graph into sub-graphs with $|E|/Q$ edges. The benefit of such an approach is

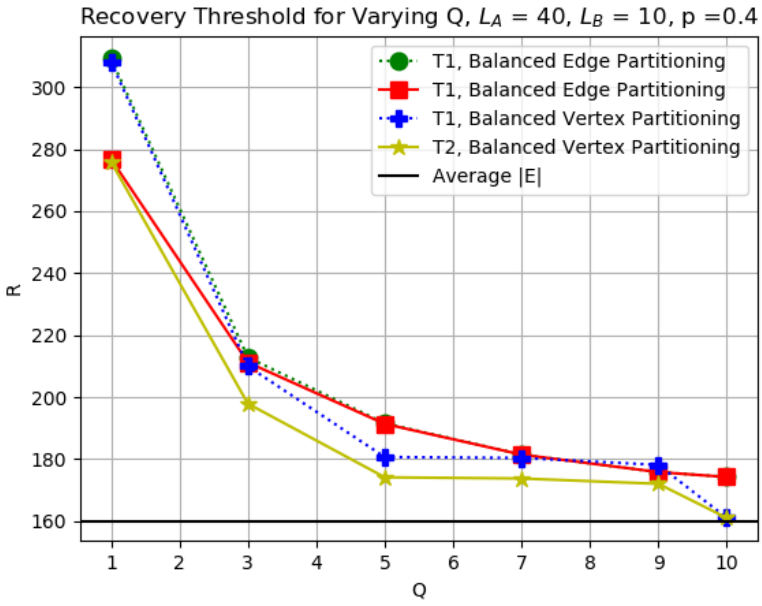
that it guarantees that $|E_q| = |E|/Q$ which would decrease $\max_{q\in[Q]}(|E_q| - \min_{i\in U_q}(d^A_{i,q}) - \min_{j\in V_q}\sum_{i\in N^q_B(j)} P^q_{i,j})$ almost uniformly. We term this partitioning as the Balanced Edge Partitioning.

Another approach is by partitioning the vertices. Without loss of generality, assume that $L_A < L_B$ and that Q divides $|U|$. We create the graph edge partition by evenly splitting all the left vertices U into groups of size $|U|/Q$ and including all the adjacent edges and right vertices connected to the chosen left vertices. This is approach is inspired by the observation in section 5.1 where we achieve the optimal recovery threshold by setting $Q = L_A$ and using this partitioning method. Additionally, this partitioning guarantees a reduction in the total number of vertices in each partition which significantly reduces the cost of encoding each partition as shown in section 5.2. We term this partitioning as the Balanced Vertex Partitioning.

We demonstrate the efficacy of both approaches by simulating their expected recovery threshold on the random graph ensemble $\mathcal{G}_{L_A,L_B,p}$. In Fig 5.1a, we present the expected recovery for $L_A = 80$, $L_B = 5$, and $p = 0.4$ across varying Q. We plot the average number of edges to indicate the fundamental lower bound. As expected, the recovery threshold decreases as Q increases for all partitioning schemes. Interestingly, we note that after a certain point GFCSA-T1 codes with Balanced Vertex Partitioning seem to outperform GFCSA-T2 codes with Balanced Edge Partitioning. Since GFCSA-T1 codes have faster encoding and decoding than GFCSA-T2 codes, finding situations where GFCSA-T1 codes provide a sufficient recovery threshold allows overall improvement in all aspects of the codes. Additionally, we see that codes with Balanced Vertex Partitioning eventually reach the optimal lower bound while codes with Balanced Edge Partitioning have a definite gap. Finally, we note that GFCSA-T2 codes with Balanced Vertex Partitioning perform the best among all the codes and partitioning methods. We see similar observations for the experiment using $L_A = 40$, $L_B = 10$, and $p = 0.4$, as shown in Fig 5.1b.



(a)



(b)

Figure 5.1: Expected Recovery thresholds of GFCSA-T1 and GFCSA-T2 codes for varying number of groups Q. Simulations are performed over the graph ensemble $\mathcal{G}_{L_A,L_B,p}$.

CHAPTER 6

Conclusion and Future Work

In this thesis, we present the novel problem of Variable Coded Distributed Batch Matrix Multiplication (VCDBMM). VCDBMM focuses on the general problem of multiplying many different matrices whose products may share matrices with no guaranteed regularity. This problem encompasses many previously researched problems of highly structured computation tasks [1–3]. While codes for these structured tasks can be applied for VCDBMM, we present Flexible Cross-Subspace Alignment (FCSA) codes that significantly improve on the recovery threshold, thereby allowing more resilience to stragglers and faster acquisition of the desired computations.

We provide two variants of FCSA codes that focus on different parts of the problem. The first variant provides a coding solution with fast encoding time and strict improvement of the recovery threshold in comparison to naive applications of previous code constructions. We demonstrate the efficacy of the first variant by showing that its improvement of the recovery threshold grows linearly with the number of matrices used in the computation. We also show how well the first variant performs on random ensembles of the VCDBMM problem. We then provide a second variant of FCSA codes which further improves the recovery threshold but requires solving a mixed-integer program to determine the optimal structure. Therefore, the second variant incurs a large online encoding complexity which can hamper the speed of computation but with large gains for the recovery threshold. As such, code optimization should be performed offline if the user knows what type of tasks will be assigned beforehand. Finally, we extend the structure of FCSA codes to allow for a trade-off between the worker's

computational complexity and the recovery threshold without affecting the download cost of the fusion node. We accomplish this by allowing certain computations to be grouped, computed separately, and combined at the worker nodes. While methods to determine optimal groupings are not yet known, we demonstrate how random grouping provides good average performance in terms of the recovery threshold.

We now discuss some possible future directions for this work. First, FCSA codes achieve a recovery threshold that is within a factor of two optimal but do not achieve the demonstrated lower bound. We speculate that FCSA codes, for many instances of the VCDBMM problem, are strictly bounded away from their optimal recovery threshold. Consider the case when $L_A = L_B = n$ and the bipartite graph is fully dense. This corresponds to the problem of finding all pairwise products in $\mathcal{A}$ and $\mathcal{B}$. Polynomial codes are proven to achieve an optimal recovery threshold of n^2 while it can be shown that the best FCSA-T2 codes have a recovery threshold of $\frac{3n^2-3n+2}{2}$. As such, one possible research direction is improving FCSA codes for highly dense graphs and providing a more general lower bound for the recovery threshold in terms of some constructs in the graph.

Another future direction is improving the optimization speed for constructing FCSA-T2 codes, as was mentioned previously. Currently, we use standard branch and cut solvers to optimize our FCSA-T2 codes which, in the worst case, can require an exponential number of arithmetic operations. Future research is required to determine whether there are properties of our optimization problem that can be exploited to improve the speed of optimization.

APPENDIX A

A.1 Proof of Lemma 1

We prove this lemma by contradiction. Assume that $\mathbf{V}_{N,K,\{u_i\}_{i=1}^N}$ is not invertible. Therefore, there exists a non-zero vector $\mathbf{c} = [c_i]_{i=1}^K$ where $c_i \in \mathbb{F}$ such that

$$\sum_{i=1}^{N}\sum_{j=1}^{u_i}\frac{c_{w_i+j}}{(x_k-f_i)^j} + \sum_{i=M+1}^{K} c_i x_k^{i-M-1} = 0 \implies \sum_{i=1}^{N}\sum_{j=1}^{u_i}\frac{c_{w_i+j}}{(x_k-f_i)^j} = -\sum_{i=M+1}^{K} c_i x_k^{i-M-1} \quad \text{(A.1)}$$

for all $\{x_i\}_{i=1}^K$ where $w_i = \sum_{r=1}^{i-1} u_r$ for $i > 1$ and $w_1 = 1$. Let

$$f(x) = \sum_{i=1}^{N}\sum_{j=1}^{u_i}\frac{c_{w_i+j}}{(x-f_i)^j} \quad \text{(A.2)}$$

$$g(x) = -\sum_{i=M+1}^{K} c_i x^{i-M-1}. \quad \text{(A.3)}$$

and define

$$h(x) = \prod_{i=1}^{N}(x-f_i)^{u_i} \quad \text{(A.4)}$$

where the degree of $h(x)$ is M. By assumption, we have that

$$h(x_k)f(x_k) = h(x_k)g(x_k) \quad \text{(A.5)}$$

for all $\{x_i\}_{i=1}^K$. Note that $h(x)f(x)$ is a polynomial of degree $\max_{i\in[N]}(M-u_i)$ and that $h(x)g(x)$ is a polynomial of degree $K-1$. Recall that $M+1 \leq K$. As such, $h(x)f(x)$ and $h(x)g(x)$ can each be uniquely determine by K distinct evaluations points. Therefore, these two must be the same polynomial. But the degrees of the polynomials can never match so it must be that $c_i = 0\ \forall i \in [K]$. Hence, we have a contradiction and the proof is complete.

A.2 Proof of Lemma 3

By linearity of expectation, we consider $\mathbb{E}[\min_{i\in[L_A]} d_i^A]$ and $\mathbb{E}[\min_{i\in[L_B]} d_i^B]$ separately. As such, we focus on $\mathbb{E}[\min_{i\in[L_A]} d_i^A]$. It is straightforward to see that $d_i^A \sim B(L_B, p)$ where $B(n,p)$ is the binomial distribution with parameters n and p. Additionally, we see that all $\{d_i^A : i \in [L_A]\}$ are independent of each other because of the bipartite structure of the graph. We observe that $d_i^A = L_B - \widetilde{Y}_i$, where $Y_i \sim B(L_B, 1-p)$, which results in $\mathbb{E}[\min_{i\in[L_A]} d_i^A] = L_B - \mathbb{E}[\max_{i\in[L_A]} Y_i]$. We thus proceed to find an upper bound for $\mathbb{E}[\max_{i\in[L_A]} Y_i]$.

Given a constant $s > 0$, we apply Jensen's Inequality on $e^{s\mathbb{E}[\max_{i\in[L_A]} Y_i]}$ and proceed as follows

$$
\begin{aligned}
e^{s\mathbb{E}[\max_{i\in[L_A]} Y_i]} &\leq \mathbb{E}\big[e^{s\max_{i\in[L_A]} Y_i}\big] && \text{(A.6)}\\
&= \mathbb{E}\big[\max_{i\in[L_A]} e^{sY_i}\big] && \text{(A.7)}\\
&\leq \sum_{i=1}^{L_A} \mathbb{E}[e^{sY_i}] && \text{(A.8)}\\
&= L_A(p + (1-p)e^s)^{L_B} && \text{(A.9)}
\end{aligned}
$$

where we input the moment-generating function of $B(L_B, 1-p)$ for the final line. By taking the natural log of both sides and fixing $s = 1$, we arrive at

$$
\mathbb{E}\big[\max_{i\in[L_A]} Y_i\big] \leq \ln(L_A) + L_B \ln(p + (1-p)e). \tag{A.10}
$$

Therefore,

$$
\mathbb{E}\big[\min_{i\in[L_A]} d_i^A\big] + \mathbb{E}\big[\min_{i\in[L_B]} d_i^B\big] \geq (L_A + L_B)(1 - \ln(p + (1-p)e)) - \ln(L_A) - \ln(L_B). \tag{A.11}
$$

We observe that $0 \leq (1 - \ln(p + (1-p)e)) \leq 1$ and that it is a monotonically decreasing function in p that attains its maximum (minimum) value at $p = 1$ ($p = 0$). Thus, we state that for $p > 0$,

$$
\mathbb{E}\big[\min_{i\in[L_A]} d_i^A + \min_{i\in[L_B]} d_i^B\big] = \Omega(L_A + L_B) \tag{A.12}
$$

REFERENCES

[1] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Polynomial codes: an optimal design for high-dimensional coded matrix multiplication," in *Proc. Conference on Neural Information Processing Systems (NIPS)*, 2017.

[2] Q. Yu, N. Raviv, J. So, and A. S. Avestimehr, "Lagrange coded computing: Optimal design for resiliency, security and privacy," *ArXiv*, vol. abs/1806.00939, 2019.

[3] Z. Jia and S. A. Jafar, "Cross Subspace Alignment Codes for Coded Distributed Batch Computation," *ArXiv*, vol. abs/1909.13873, 2019.

[4] J. Dean and L. A. Barroso, "The tail at scale," *Communications of the ACM*, vol. 56, pp. 74–80, 2013.

[5] H. Weatherspoon and J. Kubiatowicz, "Erasure coding vs. replication: A quantitative comparison," in *Peer-to-Peer Systems*, 2002, pp. 328–337.

[6] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Transactions on Information Theory*, vol. 64, no. 3, pp. 1514–1529, Mar 2018.

[7] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "A unified coding framework for distributed computing with straggling servers," in *IEEE Globecom Workshops*, 2016, pp. 1–6.

[8] S. Dutta, V. R. Cadambe, and P. Grover, "Short-Dot: Computing large linear transforms distributedly using coded short dot products," *IEEE Transactions on Information Theory*, vol. 65, pp. 6171–6193, 2016.

[9] A. B. Das and A. Ramamoorthy, "Distributed matrix-vector multiplication: A convolutional coding approach," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, 2019, pp. 3022–3026.

[10] A. Reisizadeh, S. Prakash, R. Pedarsani, and A. S. Avestimehr, "Coded computation over heterogeneous clusters," *IEEE Transactions on Information Theory*, vol. 65, pp. 4227–4242, 2019.

[11] S. Dutta, V. R. Cadambe, and P. Grover, "Coded convolution for parallel and distributed computing within a deadline," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, 2017, pp. 2403–2407.

[12] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Coded fourier transform," in *Proc. 55th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2017, pp. 494–501.

[13] Z. Lin, K. G. Narra, M. Yu, A. S. Avestimehr, and M. Annavaram, "Train where the data is: A case for bandwidth efficient coded training," *ArXiv*, vol. abs/1910.10283, 2019.

[14] H. Jeong, T. M. Low, and P. Grover, "Coded fft and its communication overhead," *ArXiv*, vol. abs/1805.09891, 2018.

[15] F. Haddadpour and V. R. Cadambe, "Codes for distributed finite alphabet matrix-vector multiplication," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, 2018, pp. 1625–1629.

[16] S. Wang, J. Liu, N. B. Shroff, and P. Yang, "Fundamental limits of coded linear transform," *ArXiv*, vol. abs/1804.09791, 2018.

[17] M. Kim, J. yong Sohn, and J. Moon, "Coded matrix multiplication on a group-based model," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, 2019, pp. 722–726.

[18] N. S. Ferdinand and S. C. Draper, "Hierarchical coded computation," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, 2018, pp. 1620–1624.

[19] H. Park, K. Lee, J. yong Sohn, C. Suh, and J. Moon, "Hierarchical coding for distributed computing," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, 2018, pp. 1630–1634.

[20] A. Mallick, M. Chaudhari, and G. Joshi, "Rateless codes for near-perfect load balancing in distributed matrix-vector multiplication," in *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 3, 2019, pp. 1 – 40.

[21] S. Dutta, M. Fahim, F. Haddadpour, H. Jeong, V. R. Cadambe, and P. Grover, "On the optimal recovery threshold of coded matrix multiplication," *IEEE Transactions on Information Theory*, vol. 66, pp. 278–301, 2020.

[22] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding," in *Proc IEEE International Symposium on Information Theory (ISIT)*, 2018, pp. 2022–2026.

[23] S. Dutta, Z. Bai, H. Jeong, T. M. Low, and P. Grover, "A unified coded deep neural network training strategy based on generalized polydot codes," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, 2018, pp. 1585–1589.

[24] S. Wang, J. Liu, and N. B. Shroff, "Coded sparse matrix multiplication," in *Proc. International Conference on Machine Learning (ICML)*, 2018, pp. 5152–5160.

[25] U. Sheth, S. Dutta, M. Chaudhari, H. Jeong, Y. Yang, J. Kohonen, T. Roos, and P. Grover, "An application of storage-optimal matdot codes for coded matrix multiplication: Fast k-nearest neighbors estimation," in *Proc. IEEE International Conference on Big Data (Big Data)*, 2018, pp. 1113–1120.

[26] H. Jeong, F. Ye, and P. Grover, "Locally recoverable coded matrix multiplication," in *Proc. 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2018, pp. 715–722.

[27] T. Z. Baharav, K. Lee, O. Ocal, and K. Ramchandran, "Straggler-proofing massive-scale distributed matrix multiplication with d-dimensional product codes," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, 2018, pp. 1993–1997.

[28] Q. Yu and A. S. Avestimehr, "Entangled polynomial codes for secure, private, and batch distributed matrix multiplication: Breaking the "cubic" barrier," *ArXiv*, vol. abs/2001.05101, 2020.

[29] B. Hasircioglu, J. Gómez-Vilardebó, and D. Gündüz, "Bivariate polynomial coding for exploiting stragglers in heterogeneous coded computing systems," *ArXiv*, vol. abs/2001.07227, 2020.

[30] T. S. Malladi and B. S. Rajan, "A computation vs communication tradeoff in distributed matrix multiplication over finite fields," in *Proc. IEEE International Conference on Communications (ICC)*, 2019, pp. 1–7.

[31] A. M. Subramaniam, A. Heidarzadeh, and K. R. Narayanan, "Random khatri-rao-product codes for numerically-stable distributed matrix multiplication," in *Proc. 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2019, pp. 253–259.

[32] A. B. Das, A. Ramamoorthy, and N. Vaswani, "Random convolutional coding for robust and straggler resilient distributed matrix computation," *ArXiv*, vol. abs/1907.08064, 2019.

[33] V. Strassen, "Gaussian elimination is not optimal," *Numerische Mathematik*, vol. 13, pp. 354–356, 1969.

[34] D. Coppersmith and S. Winograd, "Matrix multiplication via arithmetic progressions," *Journal of Symbolic Computation*, vol. 9, no. 3, pp. 251 – 280, 1990.

[35] F. Le Gall, "Powers of tensors and fast matrix multiplication," in *Proc. International Symposium on Symbolic and Algebraic Computation (ISSAC)*, 2014, pp. 296–303.

[36] C. S. Iliopoulos, "Worst-case complexity bounds on algorithms for computing the canonical structure of finite abelian groups and the hermite and smith normal forms of an integer matrix," *SIAM J. Comput.*, vol. 18, pp. 658–669, 1989.

[37] M. Gasca, J. Martnez, and G. Mhlbach, "Computation of rational interpolants with prescribed poles," *Journal of Computational and Applied Mathematics*, vol. 26, no. 3, pp. 297 – 309, 1989.

[38] Z. Jia and S. A. Jafar, "x-secure t-private information retrieval from mds coded storage with byzantine and unresponsive servers," *ArXiv*, vol. abs/1908.10854, 2019.

[39] V. Olshevsky and A. Shokrollahi, "A superfast algorithm for confluent rational tangential interpolation problem via matrix-vector multiplication for confluent cauchy-like matrices," *Structured Matrices in Mathematics, Computer Science, and Engineering I*, vol. 280, 2001.

[40] I. Gohberg and V. Olshevsky, "Fast algorithms with preprocessing for matrix-vector multiplication problems," *J. Complex.*, vol. 10, pp. 411–427, 1994.

[41] Gurobi Optimization, LLC, "Gurobi optimizer reference manual," 2020. [Online]. Available: http://www.gurobi.com

[42] "coin-or/cbc: Version 2.10.5," Mar. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3700700

[43] J. E. Mitchell, "Branch-and-cut algorithms for combinatorial optimization problems," *Handbook of applied optimization*, vol. 1, pp. 65–77, 2002.

[44] T. Achterberg and R. Wunderling, "Mixed integer programming: Analyzing 12 years of progress," *Facets of Combinatorial Optimization: Festschrift for Martin Grötschel*, pp. 449–481, 2013.